

Information Valets for Intelligent Information Access

Sofus A. Macskassy^{†‡}, Aynur A. Dayanik[†], Haym Hirsh[†]
 {sofmac,aynur,hirsh}@cs.rutgers.edu

[†]Department of Computer Science
 Rutgers University
 110 Frelinghuysen Rd
 Piscataway, NJ 08854-8019

[‡]Information Architects
 Morrocroft II
 2064 Colony Road
 Charlotte, NC 28211

Abstract

This paper introduces Information Valets (“iValets”), a general framework for intelligent access to information. Our goal is to support access to a range of information sources from a range of client devices with some degree of uniformity. Further, the information server is aware of its user and user devices, learning from the user’s past interactions where and how to send new incoming information from whatever information is available for the given task. Our metaphor is that of a valet that sits between a user’s client devices and the information services that the user made want to access. The paper presents the general structure of an iValet, cataloging the main design decisions that must go into the design of any such system. To demonstrate this, the paper instantiates the abstract iValet model with a fielded prototype system, the EmailValet, which learns users’ email-reading preferences on wireless platforms.

Problem Description

Imagine that while walking to lunch, your mobile phone rings and tells you to turn on your Palm VII and read ABC news because there is an important news story regarding a stock in your portfolio; or on your way to a meeting, you receive a message on your pager telling your the meeting has been moved to another location. 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. Some messages would likely be relevant to you at that particular point in time, while others would not. For example, it would be desirable to have a single, centralized location where all information is gathered. A software agent residing at that location could then forward each piece of information to where the user would like to read it. Ideally, this agent would work in a way much akin to a secretary, knowing where the user is and knowing which information to send to the user on a particular device (if the user has more than one wireless device). We call such an agent an Information Valet, or iValet, in that it provides “valet”-like functionality to help users manage information, by adapting itself to the user’s interest.

This paper first describes the structure of an Information Valet, followed by the questions that must be answered in the design of any such system. It continues by describing an instantiation of an iValet, the EmailValet (Macskassy, Dayanik, & Hirsh 1999), and experimental results on applying different learners to the data gathered so far.

Information Valet Framework

The goal of an iValet is to provide access to information from a range of client devices in intelligent and useful ways. We have identified a set of different dimensions that are important for the design of an iValet. These dimensions center around four components; the user, the device(s), the presentation and the information. Each component imposes certain

constraints and behaviors on the overall design. This section describes in more detail each of these main components.

User

The user is at the center of this framework; the iValet is here to serve the user in the best way possible, making user satisfaction and usability very important. There are a range of decisions that impact how the system interacts with the user, each being important for the overall feel of the system. There is the *user model* which represents what is known about the user interests; the *user context* represents information about where the user currently is, what the user is doing, current tasks, action items, etc.; there is the *user feedback* dimension which interacts with the user and feeds the *learning* component information to help updating the user model and present the information more intelligently at the next user interaction. Each dimension is described in more detail below, with descriptions of the range of possible values each dimension may take on.

User Context What is important prior to a business meeting might not be as important if you are on vacation; what is important on the way home (ie, pick up groceries) is not important in another situation. The necessary context is somewhat dependent on the overall application and domain of the iValet, however some of the salient features that are of general importance include:

1. **location:** Where is the user currently? At work, in the city, on the road, in another country? The location could either be a discrete set of tokens, or more exactly a GPS point. If neither of these are available, approximations or substitutes could potentially be used.
2. **state:** What is the state of the user (Takkinen & Shahmehri 1998)? Busy, on vacation, at lunch, normal workday, in a meeting? This type of information might have to come directly from the user, or may be available through other means, such as from an on-line calendar; many people have a calendar that list their meetings and other important dates and times, including a class schedule, trips, vacations. This information could be of great use by the iValet to decide upon the importance of information.
3. **availability:** Is the user currently available to the iValet through one or more listed devices? This could either be logged-in, a turned-on pager or mobile phone, an intercom accessible by the iValet, or other means available through current technology.

User Model The user model represents what is known about the user’s interest and possibly more in-depth knowledge about the user. Our approach takes a minimalist approach where we refrain from creating complex knowledge based structures about the user, but rather focus on what can be learned implicitly by user feedback, the information itself, and possibly

contextual information available to the iValet. Others (Pazzani & Billsus 1997; Billsus & Pazzani 1999) have pointed out that users have at least two levels of interests; long-term and short-term. What information should the user model store, and how? Furthermore, if something is already known, it might not be of as much interest as something new (Billsus & Pazzani 1999).

1. **short-term:** What is the user currently interested in. This could quickly change as the user either gets distracted or branches off based on previously seen information. This type of user model must quickly adapt to these changes.
2. **long-term:** A more stable interest of a user might be a hobby or work-related. This type of interest stays fairly stable over time, and reflects information that the user is always interested in. The model should be capable of tracking a gradually change this interest depending on what the user currently reads.
3. **hybrids:** A hybrid model endeavours to incorporate interests of both kinds. The user model can either keep both types of information together and not distinguish them, making it hard to update or, as seen in NewsDude (Billsus & Pazzani 1999), be a meta-user model that keep the two models separate but give a composite value to new information based on each model.
4. **already known:** If a user has already read about this topic, this information might not be of interest unless new information as also given. This is of particular use when presenting information to a user so that new information gets more weight than already-known information.

Another distinction of information in the user model is whether to use only data from within the information items themselves (content-based), or whether to use collaborative information, taken across multiple users or information items:

1. **content:** Information is based on a per-instance framework, and the user model can only predict or represent information within a single instance. A drawback to using this model by itself is that new instances are hard to categorize if they do not fall within, or close to, an already seen category.
2. **collaborative:** Information is taken from a range of users (Lashkari, Metral, & Maes 1994), or a range of information items (Shardanand & Maes 1995). Using this type of model, it is possible to categorize completely new instances for one user if at least one other user, with similar interests, has seen information of this type. This circumvents the problem of the content based model, but has the drawback that a lot more information needs to be seen in order for the model to perform well.
3. **hybrid:** A hybrid approach makes use of both types of information (Basu, Hirsh, & Cohen 1998). By mixing information from both models, albeit very carefully, it is possible to improve the model and get better results than using either one.

The second, and more subtle, dimension of user models is that of dealing with multiple devices. Since each device has different capabilities and constraints, what is important on one device is not so important on another device. The user model needs to take into account not only the user interest, but also how this interacts with each device. For example, one device might be very costly to send data to, but also have many capabilities (bandwidth, graphics, etc.), while another device is slow and cheap. Where should information be sent? In the iValet approach, we plan to take the hybrid model to the next step and include a model for each of the user's client devices; these extra models will help the iValet

decide where a given information instance will be preferred to be seen, as well as which part of the information to send. It could even lead to sending a short one-line message to a pager or mobile phone to tell the user to turn on a palmtop and read email there, if the email is too big to read on the mobile phone/pager, but important enough to spend time and money to read it on the palmtop rather than wait until later.

User Feedback How the user interacts with the system is important; equally important is what part of the interaction can be used for learning. This part, the *user feedback*, can be used by the learner to update the user models which in turn are used to present information to the user. Feedback can be range of interactions from the user and can either be implicit or explicit, based on the interface. There is a very fine line between asking for clarification and being in the way of the user. It is of utmost importance to be useful without being in the way. However, sometimes the system could greatly benefit from being able to ask the user for clarifications. Our approach in the iValet is to be minimal and try to get as much information as possible with the least amount of intrusion, while still letting the user give more feedback, if so desired. This interaction paradigm has also been used by NewsDude (Billsus & Pazzani 1999). We have identified a list of possible implicit feedback:

1. **read item:** When reading an information item, the user is obviously interested, if only for a short period of time, in that particular item. The further it is read (if this can be tracked), the more of an interest this is. The learner can use that information item as a positive example *for that device*. There is a subtle point here; if an item is read on a particular device, then it is most likely preferred to be read there. However, it also gives a general interest for that particular piece of information.
2. **delete item:** If the item is deleted, then the learner can use this action as implicit feedback for a negative instance. Unless there a multiple copies available or the item was forwarded to another device, this action should be used as a negative for all devices.
3. **move to device:** If moving a particular piece of information to a device (eg, download to a PDA or send email to a pager), then this gives decreases weight on the device from which the information was forwarded, while increasing the weight on the receiving device.
4. **skip item:** If information was listed and a piece of information was skipped over to read another piece of information below it, then this gives the feedback of *relative ordering* between unread information above the selected information and the information selected.
If relevant, the system could also be able to deal with explicit feedback from the user, and be able to *explain* (Billsus & Pazzani 1999) why certain decisions were made. The explanations could come directly from how the user models were used to decide on how interesting a piece of information was. Due to this, the explicit feedback as well as the explanations are very dependent upon the chosen user models and learners used. We have identified explicit feedback that is general enough to be included in most systems:
 1. **strengthen:** The decision was good enough that the user felt it should be strengthened (ie, if one or more tokens were used as examples for why this piece of information was rated this way, increase the weight of those tokens).
Note: this is not the same as agreeing with the decision, in which case nothing should happen.
 2. **bad:** The decision was bad, and whatever was used to make this decision should be weighted accordingly (ie, if a piece of information was used as a positive example to make this decision, decrease the weight of that example).

3. **remove:** The information used for this decision was so bad and made no sense to the user. In fact, this was so off base that the user asked the system to totally remove (part of) the information used for this decision.
4. **add:** This way the user can add information to the user model, if so desired. This is not feedback to a particular decision, per se, but having the user add information based on personal knowledge of his or her own interests.

Learner Both the learner used and what is to be learned is important. From related research (Basu, Hirsh, & Cohen 1998) in dealing with multiple sources of information in varying domains, we have identified three distinct types of learning possible on a set of data. This learning is dependent upon the interface, the learner capabilities, the data itself and the domain and application thereof. Each learning type represent the amount of information given back to the user from very precise to very vague. If the data is labeled as binary classes, then the learner can only predict into these classes; however, if the data supports more complex interrelational information, then the learner can use this. On the other hand, should the final outcome of the learning only need to be binary, then the learner should be able to output classifications that fit within that goal. The types of learning are:

1. **scoring:** This is the most precise and detailed information that a user can get. Each piece of information has a score that precisely represents how important or interesting this piece of information is to the user (based on the user model and learner, of course).
2. **sorting:** Many times the learner cannot give a good score to a piece of information, for a variety of reasons. In this case, the learner can fall back to a more modest and safe bet; namely do a relative ordering of information. It might not know how interesting a piece of information is, but it does know that it is *more* interesting than another piece of information. Using relative ordering, the learner can present the data to the user in an ordering from most to least interesting.
3. **labeling:** The least informative and most safe information the learner can give is just a plain labeling into a discrete set of labels; in the most basic case, it can label a piece of information as either interesting or not-interesting. It does not commit to actually ordering any pieces of information within each label.

The final outcome can be less detailed than the information given by the data, though it is not possible to take data with binary classes and output an absolute scoring.

User Trust For any adaptive interface/system, it is important for the user to learn to trust it. While this is not a value or feature that can be gotten, it is still important to keep in mind that the user has an experience whereby a relationship can be grown between the user and the interface such that the user sees how the adaptation learns about the user and gets better at identifying information that the user wants (Maes & Kozierok 1993; Foner & Maes 1994).

Devices

The second component of an iValet is that of the device. An iValet can make use of multiple devices, each having different characteristics and capabilities. For example, the user could have a pager that can send and receive email, is always on, but has a small and limited screen; a palm device that has graphics and a bigger screen, but is only on when the antenna is raised; and a mobile phone that can receive short one- or two-line messages through SMS. Each device has certain capabilities, contexts and constraints and by watching how the user makes use of each device, the iValet can learn when and

where to send new information. There are three main areas of information that are important to know about a device.

Device Connectivity The connectivity issue is subtle in what should be learned about that particular device. For example, if the device is always on, then the important thing to learn is whether the send a new piece of information or not. This translates into a binary classification task. However, if the device is intermittently connected, then the important thing to learn would be either a relative or absolute ordering of new information since last time the device was connected. We list below the important features for a device:

1. **always connected:** If the device is always connected, then the decision for an new piece of information is whether to send that information or not. Depending on the actual learning dimension, this translates into either an absolute threshold or a binary classification.
2. **intermittent connectivity:** If the device is only connected from time to time, which is the case laptops with wireless modems, or with palm devices like the Palm VII that are only connected with the antenna is raised, then the decision is not whether to send information but rather how to list new information when the user connects.
3. **push/pull capable:** If the device is capable of receiving information in push (eg, as a page, a phone-call, an email), or if the information has to be pulled by the device (eg, as a web-browser style interface or email pop-interface). If the device is push-technology enabled (a phone can receive SMS messages all the time with no user interaction), and if it can let the iValet know when it is connected, then then iValet can push important information to the device as soon as it comes online. A device which can only pull, could be a laptop; when dialing in using a wireless or hard phone line, then the use can use a web-browser and request information from a web-server.
4. **email-capable:** Is the device available through email? If so, the iValet has a clear way to send (or push) information to the device. If the device is intermittent, it can still be sent email and if the information is deemed important enough, the user can be sent a page or SMS (on the phone) to be told to turn on the device.
5. **bandwidth:** The bandwidth of the device; this could affect what (or how) information to send to the device, either due to the amount of time or the amount of cost (in money or battery) for sending the information to the device.
6. **wireless:** Is the device wireless? It is a feature or characteristic of the device, though it is unclear whether this piece of information is important.

Device Capabilities What are the capabilities of the device in terms of presentation and storage? Maybe this is a storage-less device, in which case information can not be sent there to be stored; or if the information is all audio or graphics and the device only has text, then there is no reason to send the information; unless it can somehow be transformed. We envision that, pending available technology, the iValet could transform audio into text, text into audio, and using OCR convert graphics into text. If none of these technologies are available or relevant, only limited information could be sent (to a text-only device); this information could include size of the information, maybe an originator/author, name of a file, etc. A list of capabilities for a device include:

1. **max item size:** What is the biggest message, in bytes, that the device will accept. Most pagers, phones, and palm-devices have a preset maximum limit on how much data they can accept in one message. If the information is bigger, it will have to be sent over multiple messages.

2. **text I/O:** Is the device able to receive (and send) textual information. If so, it would also be important to know the maximum screen size in terms of columns and rows. This could affect how much information (and maybe how to format it) that should be sent to the device.
3. **audio I/O:** Is the device able to receive and/or send audio information? If it can receive audio information, at what quality? This means that a voice-message could be forwarded to the device rather than to try and do a speech-to-text conversion first. If the device can send audio, the iValet must have a way to receive this audio information.
4. **graphics:** Is the device graphics capable? If so, can it accept color? At what bit-depth? Again, screen size is important for this so that the iValet can transform graphics to be only big enough to be sent to the device.
5. **local storage:** Does the device have local storage available? If this is the case, then the user should have the option of sending information to the device for local storage. The iValet can then keep track of what information has been sent to the device and keep track of how many copies are floating between the user's devices. Ideally, the iValet should then also be notified somehow if (when) the information is removed from the device.
6. **processor speed:** How powerful is the device? It is a feature or characteristic of the device, though it is unclear whether this piece of information is important.
7. **infrared ports:** With the advent of PANs (Personal Area Networks), it is now feasible for devices relatively close to each other to communicate, either through radio or infrared. This could increase its functionality; for example, if there is a printer nearby, the device could print out data and information could be sent to it that could not normally be presented on its screen could now be printed out.

As devices become more powerful, these capabilities should also get bigger, though there by virtue of the small devices is only so much screen real estate possible. An important fact to note is that while audio is good, it is not always the best medium. Sometimes text (and/or graphics) is needed.

Device Context The final piece of information about a device is its context. the iValet should have access to as much contextual information about a device as is relevant. This is a list of device context information:

1. **battery life:** The current battery life of the device; if the iValet knows about it, it can decide accordingly which, and how much, information to send to the device.
2. **currently connected:** If the device is capable, it should let the iValet know when it becomes connected. The iValet can use this information to keep track of currently available devices through which it can contact the user.
3. **price per byte:** If relevant, how much does it cost the user to send information? In the current world, there are many options whereby you get an amount of data for a fixed price after which you pay per byte. If the iValet knows about this, it can redirect information accordingly (if it knows that the pager is almost at its quota, it might send it to the phone or palm device; the user has the option of sending the information to a specific device if desired). This predicates that the iValet has a way to find out exactly how much data has been sent to (from) the device.
4. **available to user:** Is the device currently available to the user. If this device is online, then more than one user might have access to it, or a mobile phone might be used by a spouse or friend. This is a feature of the device, however it does seem infeasible to get this information without being intrusive. We have left it in here for completeness sake.
5. **remaining storage:** If the device has local storage, how much space is available?

Information Items

The final component for an iValet is the information itself. The obvious choice is to treat each domain either as a separate domain with a special learner and/or transformation to deal with that domain or treat all information pieces as one message of all text. In the iValet approach, we plan on transforming all pieces of information, regardless of domain, into a homogeneous set of information items with a standard set of features; A feature is empty if it is not relevant for the original domain. By doing this, we hope to be able to learn about general information and treat all information the same; making domain a feature of the information item, enables the learner to make use of that information if it is relevant. We list here a sample set of domains:

1. **mail:** A mail item is an incoming mail message. The salient features in a mail message should include the sender, the receiver(s) (to and cc), the subject, date, length (potentially two numbers: the textual size and the overall size, if the email contains binary attachments), and body of the mail. The mail domain also has contextual information regarding history of email. Contextual features we consider are the last time the user got (sent) an email from(to) the sender and the last time the user got(sent) an email from(to) the sender on the particular subject.
2. **news:** There are two types of news information; that from a standard newsgroup, and a news story available online. Both types have an author, a body, a date, and overall size of the item; however, the newsgroup item also has a subject and a list of newsgroups where it was posted, while the online news-story has an associated press, a possible abstract and a possible list of images. The contextual information in both cases would involve a past history in that the newsgroup item could be an ongoing discussion and a news-story could be a follow-up of a previous item.
3. **stocks:** Stock quotes are very specific informational items, which includes information on the current stock price, volume trade, and past history of that price. More subtly, stock quotes can have indirect information from news stories that mention the stock (or company).
4. **files:** One important aspect of an iValet is that it should provide access to information; not only new information that is sent to the user or is of interest to the user, but also information that the user needs, regardless of its type. In our own use of the EmailValet described below, we have found that access to our files is very convenient. This is a special case of information; in the current framework, the iValet will make the files available but there are no plans to have learning take place in this domain.
5. **web:** The final domain considered in this paper is the web. The web domain is wide open and contains a lot of information that could be of interest to the user. As with files, the user can explicitly request a particular web-page; we are also considering other options where the iValet can browse the web and let the user know if any web pages are deemed to be of interest to the user. Regardless, all viewed web-pages can be used by the learner as a positive example of information that the user is interested in.

EmailValet

We have taken the iValet paradigm and instantiated it into a simple iValet which we call the EmailValet. The EmailValet is a fully functional iValet with much of the functionality described above; it is simple in that it currently only works with one device of which it has little to no knowledge.

The remainder of this section describes, using the iValet framework, the specifics of the EmailValet followed by a section describing the implementation.

User

The EmailValet makes use of much of the information described in the user component part of the iValet framework; for the *user context*, it keeps track of the users online context, namely whether the user is currently online, how long the user has been idle, and how long ago the user last read email online. This contextual information is used as a substitute for the user's location; we had no easy access to that and this seemed a reasonable, if primitive, way of at least gauging if the user had other current means of reading email.

For the *user model*, we do not distinguish between a long-term and a short-term interest; the user model consists of the users' online context, the email and its features, and the email context. The complete information is all text.

In order to get *user feedback*, we had to tweak the user interface a little; there was no easy way to let the EmailValet know when a message was read on the pagers. In order to get this information, the interface was created using a request-response interaction, where the EmailValet would first forward the headers of the email. The user had to explicitly request the complete email to get the body. Using this feedback as a **read item** feedback, we were able to learn from the user which email was important enough to be sent to the pager.

Currently there is no online *learner* used; we have done a series of offline experiments using different well-established learning methods to see if one learner was better than others; these experiments are discussed in the later part of the paper. Due to the nature of this learning task, we were only interested in labeling data as either **Forward** or **NotForward**.

Devices

The EmailValet is built on top of three RIM 950 pages with service by BellSouth Wireless Data. The RIM device is a small pager with a 'qwerty' keyboard and an 8 by 30 textual screen. The services available include sending and receiving text messages both through BellSouth's proprietary interactive pagingSM network, as well as to and from arbitrary internet email addresses.

For *connectivity*, the pager can be assumed to always be connected, though the user has the option of turning it off. Because it is capable of receiving email, it is *email-capable* and allows for *pushing* information unto the device.

The *capabilities* of the device includes a max message size of 16Kb, it is text only, and has local storage (of saved emails) in the order of 1Mb.

We had no device contextual information available to the EmailValet.

Information Items

The EmailValet primarily tries to learn about the users' email reading behavior; it makes use of all features described with respect to the email domain. Due to the interface and the fact that the EmailValet resides on the Rutgers mail-server, it also supports limited file access plus textual lynx-like web-browsing (although we do not discuss these service further in this paper).

One thing we learned from our use of the EmailValet was that integration of information was very convenient and a big plus in the usability of the EmailValet. This integration involved identifying and enumerating all URLs in emails, files and web-pages such that the user had an easy way of browsing URLs regardless of the initial format of the information item. We realize that this is very similar behavior to most of the major email readers.

Implementation

The EmailValet consists of an agent residing on the Rutgers mailserver, which filters incoming email and decides what to do with it. There are two types of incoming email that are

Table 1: Dataset properties

User	Size	% of messages forwarded in		
		full set	first $\frac{2}{3}$	last $\frac{1}{3}$
AD	2298	27.08%	25.85%	29.50%
HH	11034	26.86%	26.97%	26.73%
SM	1675	16.78%	16.65%	17.03%

treated differently; an email coming from the pager is taken as a "command" to the agent; all other email is a data point and the headers are forwarded to the pager. When sending the headers to the pager, a Reply-To field is added going back to the mail-agent (the user's own email address), this means that all replies from the pager will go back to the mail-agent for tracking. The user can either reply back to the sender or ask the agent for the complete body. When replying back to the sender, the agent again repackages the headers so that the email looks like it came from the rutgers account and not from the pager. This has the effect of hiding the pager to the outside world, and making the agent almost transparent to the user (except for having to ask specifically for the body).

Using this agent, we have been able to track three users over a period of more than nine months, and with a total of more than 15000 non-command email messages.

Experiments

We carried out some experiments with nine months worth of data gathered from the authors. From the user logs, two sets of email messages were created for each user, one containing those messages that the user chose to see, and a second that the user did not choose to see. We considered the task of learning to distinguish these classes, to learn which messages should have their full body sent to their user's pager. The experiments we report here were performed "off-line", evaluating six different learners on this binary classification task.

Learning Algorithms

We tested using a range of standard text categorization algorithms, all of which performed comparably; for this reason we will only report the results from Naive Bayes (Domingos & Pazzani 1996). We used a publically available version of this learner (part of the Rainbow (McCallum 1996) package).

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.

Data

Table 1 shows the amount of data obtained for each user, and the proportion of email that each user requested forwarded. Since data are chronological in nature, a user's email data were divided into two contiguous segments. The first two-thirds of a user's data were taken as the training set for all learning methods, and the last one third was taken as the test set for each method.

Evaluation

The aim of EmailValet is to forward the relevant messages to 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. The recall of a classification method is the proportion of all truly positive data that are labeled as positive. In our case, the precision corresponds to the proportion of messages that were labeled as forward

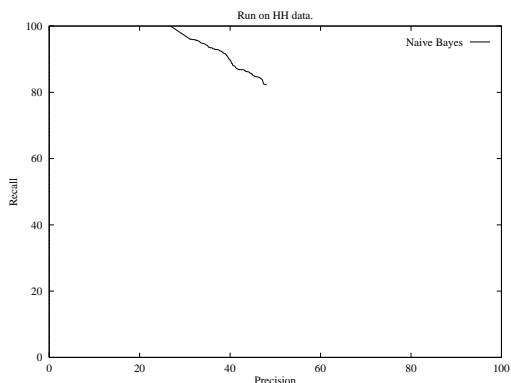


Figure 1: Recall vs. Precision for Naive Bayes on HH data.

that really should have been forwarded, and the recall corresponds to the proportion of messages that should have been forwarded that were labeled forward.

Results

Figure 1 shows a sample precision/recall graph; it allows certain conclusions to be made about the use of learning methods in this domain. 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). In all cases, for all methods, this default figure is met or, most often, surpassed, demonstrating that even the less-successful methods still achieve some modicum of success on this task.

We did a second set of experiments to find out the effect the online context and the email context had on overall performance. We ran the learner on three sets of data; one with all features, one where all online context features had been removed, and one where all email context features had been removed. We then compared the precision/recall curves on the three runs. Figure 2 shows a sample run of this experiment. Unfortunately, against our hypothesis, the contextual features had little effect on the overall performance and we plan to do further studies to find out why.

Prospects for the Future

In this paper we introduced a new framework, the Information Valet (iValet), to be able to intelligently access information using a range of different client devices. We described the main components that can be used by this framework and the specifics of how to use them. We gave an overview, for each component, of a range of possible values and types of information that could be useful and discussed issues of design and its effect the overall functionality of the iValet.

We continued by describing the EmailValet, an iValet instantiation, using the iValet framework methodology; we went through each major iValet component and which part of it was used by the EmailValet.

In the short term, we would like to understand which of the iValet components and features can be used to improve performance of the EmailValet. 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 the EmailValet across a range of wired and wireless platforms and with a wider range of information sources. We are currently looking into making use of a Palm VII device as an iValet, using a web-interface, giving the user access to email, files and the web. We would also like to expand the range of interaction modalities that a user can have with the iValet. Currently, the EmailValet only has one implicit feedback; we'd like to extend it to a wider range of modalities

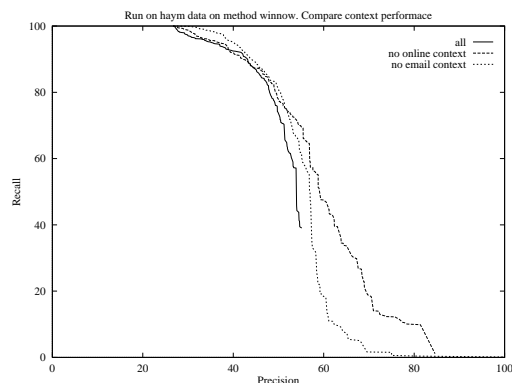


Figure 2: Effects of contextual features on HH data

as described in the user feedback section of the iValet user component.

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.

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

- Basu, C.; Hirsh, H.; and Cohen, W. W. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*.
- Billsus, D., and Pazzani, M. 1999. A personal news agent that talks, learns and explains. In *Proceedings of the Third International Conference on Autonomous Agents*.
- Domingos, P., and Pazzani, M. 1996. Beyond independence: Conditions for the optimality of simple bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning*, 105–112.
- Foner, L., and Maes, P. 1994. Paying attention to what's important: Using focus of attention to improve unsupervised learning. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. Brighton, UK: MIT Press.
- Lashkari, Y.; Metral, M.; and Maes, P. 1994. Collaborative interface agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.
- Macskassy, S. A.; Dayanik, A. A.; and Hirsh, H. 1999. Emailvalet: Learning user preferences for wireless email. In *Proceedings of Learning about Users Workshop, IJCAI'99*.
- Maes, P., and Kozierok, R. 1993. Learning interface agents. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 459–465. Menlo Park, CA: AAAI Press.
- McCallum, A. K. 1996. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>.
- Pazzani, M., and Billsus, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning* 27:313–331.
- Shardanand, U., and Maes, P. 1995. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the CHI-95 Conference*.
- Takkinen, J., and Shahmehri, N. 1998. Are you busy, cool, or just curious? – cafe: A model with three different states of mind for a user to manage information in electronic mail. In Lundgren, P. A., ed., *Human IT*, number 1 in 98. ISSN 1402-1501.