Integrating Rich User Feedback into Intelligent User Interfaces

Simone Stumpf, Erin Sullivan, Erin Fitzhenry,
Ian Oberst, Weng-Keen Wong, Margaret Burnett
Oregon State University
Corvallis, OR
{stumpf, sullivae, fitzheer, obersti, wong, burnett}@eecs.oregonstate.edu

ABSTRACT
The potential for machine learning systems to improve via a mutually beneficial exchange of information with users has yet to be explored in much detail. Previously, we found that users were willing to provide a generous amount of rich feedback to machine learning systems, and that the types of some of this rich feedback seem promising for assimilation by machine learning algorithms. Following up on those findings, we ran an experiment to assess the viability of incorporating real-time keyword-based feedback in initial training phases when data is limited. We found that rich feedback improved accuracy but an initial unstable period often caused large fluctuations in classifier behavior. Participants were able to give feedback by relying heavily on system communication in order to respond to changes. The results show that in order to benefit from the user's knowledge, machine learning systems must be able to absorb keyword-based rich feedback in a graceful manner and provide clear explanations of their predictions.

Author Keywords Machine learning, user feedback.

ACM CLASSIFICATION KEYWORDS H.5.2 [Information interfaces and presentation (e.g., HCI)] User Interfaces: Theory and methods. H.1.2 [Models and Principles]: User/Machine Systems: Human information processing, Human factors.

INTRODUCTION

Many intelligent user interfaces attempt to adapt to a user's needs based on the user's history of interaction. One increasingly common approach being brought to intelligent user interfaces is machine learning, in which the system learns new behaviors by examining usage data.

Traditionally, machine learning systems have been designed and implemented off-line by experts and then deployed. Recently however, it has become feasible to allow these systems to continue to adapt to end users by learning from their behavior after deployment. Interactive email spam filters, such as in Apple's Mail system, are prime examples.

A common problem of intelligent user interfaces that base predictions on the usage history is that training data during the initial start-up phase is sparse. Consequently, the aim is to improve accuracy quickly to an acceptable level from very few training examples. Similarly, reacting to changes in classification by the user, usually known as concept drift [10], needs to be swift and be based on only a few examples or corrections.

Also, how are such corrections made? This is a second problem: the norm for machine learning systems that take user feedback is to allow the user to indicate only that a prediction was wrong or to specify what the correct prediction should have been. This is just a glimpse of the rich knowledge users have about the correct prediction.

We believe that addressing the second problem may help to make headway on the first problem as well. Our previous work [27] has shown that machine learning systems can explain their reasoning through keywords in a way that is understandable to users, and that in turn, the users can make corrections to the reasoning. We found that the majority of feedback concerned reweighting keywords and selecting different keywords. Participants were more accurate than the machine—but they were not perfect, and occasionally made mistakes. This demonstrates both the potential viability of allowing users to correct system reasoning, and the likely pitfall of rich user feedback introducing errors into the reasoning, against which the system needs to guard.

Machine learning approaches that take rich user feedback into account are still in their infancy, and there are many open problems requiring investigation. For example, how to incorporate rich user feedback into machine learning algorithms, theoretically and practically, needs to be explored. Also, data is needed to evaluate more fully how effective these approaches are for classification, especially early in training and based on real user feedback. Finally, the usability of such approaches and their effects on users' behav-

ior needs to be studied. Obtaining feedback, incorporating it into learning algorithms and communicating its effects pose very challenging problems alone, and even more so when combined in an intelligent user interface. We are interested in exploring rich feedback approaches and their potential in improving both machine learning and user interaction.

To help provide some answers to these problems, we developed an intelligent email system that assisted a user in classifying emails into appropriate folders. In this paper, we report on an experiment that allowed users to help guide the system through adding and deleting keywords, and changing weights on keywords in emails. The predictions and changes to these predictions resulting from the user feedback were communicated by the user interface on the fly. We aimed to answer the following research questions:

- *What are the effects of user feedback on accuracy, especially when few training examples are available?*
- *What are the effects of user feedback on users: How is feedback given? What are the perceptions of communications between users and the system?*

THE EXPERIMENT

Experiment Set-up

We recruited 43 undergraduate and graduate students with fluent English-speaking skills for this experiment. All had experience using email, but none had Computer Science backgrounds. Each completed a background questionnaire with gender, GPA, major, years speaking English, and years of email experience. We obtained logs of 30 participants'[1] interactions with the experimental system as they worked with the system to classify email into folders.

The emails being classified consisted of a pool of 1151 email messages, which we considered to be the contents of a user's inbox. The *training set* consisted of 50 of these emails (ten emails each in five folders) along with their folder assignments, which were used to train the classifier initially. Fifty of the remaining emails were chosen randomly to form the *feedback set*, which were presented to participants in the main task. The *test set* consisted of the remaining 1051 emails from the original pool of emails.

We used the publicly available Enron email data set as a basis for collecting data. For this experiment, we selected nine folders containing at least ten messages. In our previous study, we had used four of these folders (Personal, Resumé, Bankrupt, Enron News), which were from user *farmer-d*. However, the Resumé and Bankrupt folders do not contain a large enough number of emails needed to train, obtain feedback and evaluate the classifier. Therefore, we combined the resumes folder from user *kaminski-v* and *farmer-d*. We also added a new folder consisting of emails from the large Systems folder of user *lokay-m*. Thus, five folders for classification were Personal, Resumé, Bankrupt, Enron News, and Systems. Finally, four additional folders (Congratulations, Floorspace, Surveys, and Enron Travel Club) from users *beck-s* and *lokay-m* were used to simulate real-world folder and filing complexity. None of the emails in the feedback set belonged to these four additional folders.

Before the main task, we introduced the participants to the basic mechanisms of providing feedback to the system. Participants were also given some time to familiarize themselves with the contents of the email folders, each of which contained several emails that had already been filed by the original user at Enron. The tutorial lasted 20 minutes. After the tutorial, the screen was cleared and participants were given 50 new emails (i.e. the feedback set) in a randomized order for the main task, which they had 40 minutes to complete.

For the main task, each participant was asked, first, to improve the system's sorting ability by correcting folder assignments and by adding, removing, and increasing the weight of ("voting up") keywords, and, second, to file correctly classified emails away. Participants were not required to work with all 50 emails within the time limit.

Finally, participants completed a post-session questionnaire asking for subjective ratings of mental effort, time pressure, overall effort, performance success, and frustration level, based on standard NASA TLX questions [16]. It also asked their level of understanding of how the system worked, the ease of feedback to change system behavior, and the level of trust they had in the system. For each rating (5-point Likert scale), participants could also give additional comments to explain the score. We also provided an opportunity for participants to tell us what would improve their ability to verify and correct the sorting suggestions.

Email Program

Several forms of obtaining user feedback have been explored in the literature, including natural language [4] and feature-value pairs [22]. A large body of work employing user feedback falls under the topic of *programming by demonstration*, which enables a system to learn a procedural program interactively from user demonstrations [12, 20, 23]. For example, the process might be ordering an item from an on-line web-site. With some exceptions [24], the user is not usually given the opportunity to provide feedback about the reasoning behind the learned program.

Previous work [10, 17] has also shown that explaining why certain outcomes happened, based on user actions, can contribute positively to system use. Similarly, it has been shown that highlighting the relationship between user actions and ensuing predictions can influence user preference [3].

Building upon this background, we devised the "E-mazing" email program. It mimicked basic features in commonly used email clients such as Mozilla Thunderbird. It also pro-

[1] 13 data files were lost due to data corruption issues.

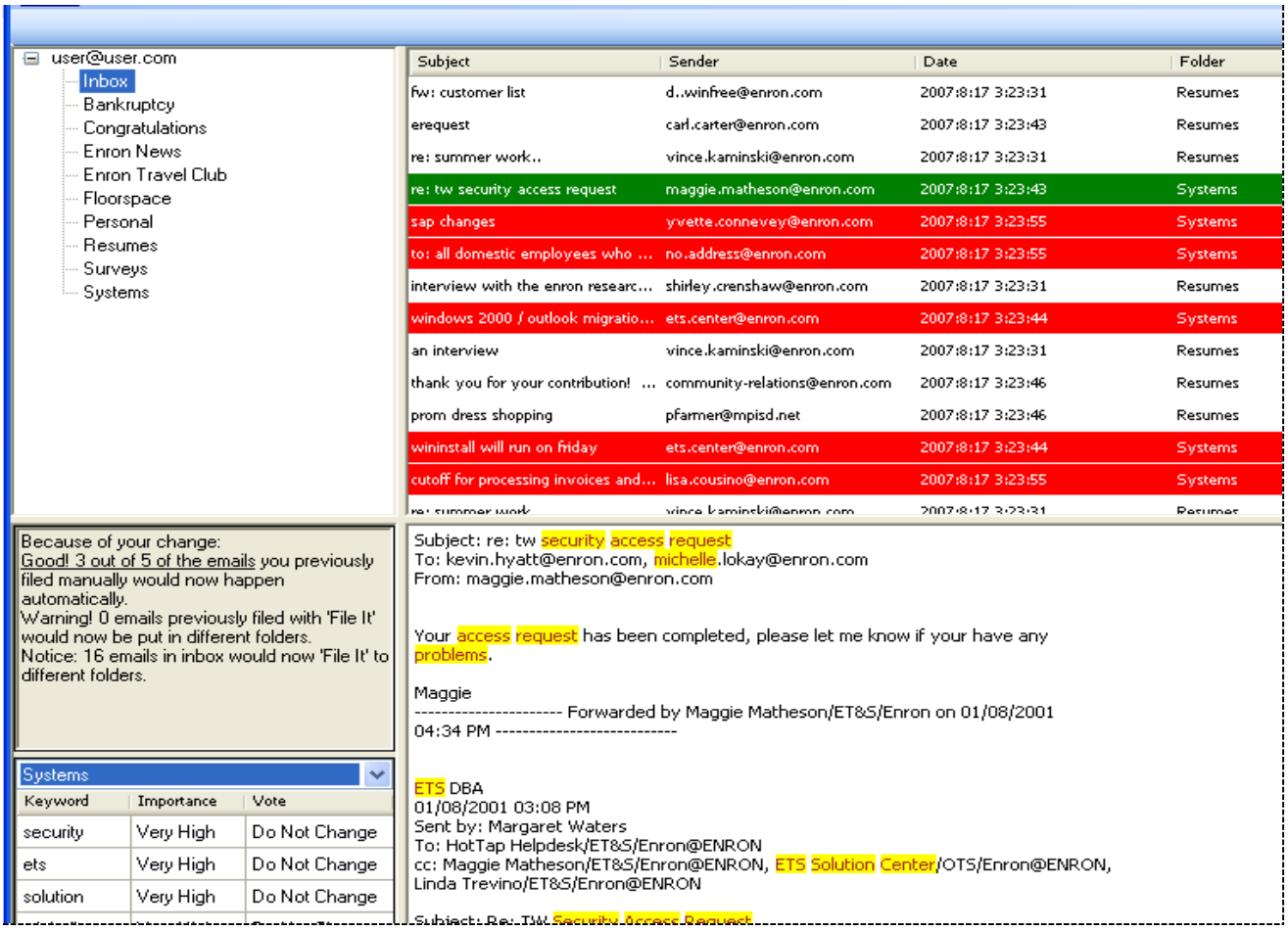

Figure 1. A partial screenshot of the E-mazing email program.

vided explanations of the classifier's reasoning about what emails belonged in what folders (using the "keyword-based" explanations technique of [27]), and provided ways users could give feedback to the classifier to improve its predictions.

To explain its reasoning, as shown in Figure 1, the feedback panel (bottom left) displayed the top ten keywords for the selected email to explain why it had been classified in the predicted folder. These keywords were also highlighted in the email message displayed in the preview pane (large pane at the bottom),

The feedback panel was the main way through which the user could give feedback. It allowed participants to select words in an email that should be treated as keywords, to make previously selected keywords be ignored, and to adjust the weights of each keyword. The keywords displayed were the ones that had been learned for each folder. Initially the folder, displayed in the dropdown menu at the top of the panel, was chosen for the current email message by the classifier. If the participant saw the folder choice as incorrect, s/he could choose a different folder, which then switched its keyword list and highlighting to those keywords important to the new folder.

In E-mazing, some keywords can be given more influence than others. For example, the keyword "resume" might be weighted heavily for the Resumes folder. The weights ranged from *very low*, *low*, *medium*, *high*, to *very high*. Participants could tell the classifier to change the weight by adjusting the *vote slider*. This slider told the classifier to increase the weight of the keyword by the amount indicated, which ranged from "Do Not Change" to "Increase a lot!!". Once a participant was satisfied, they pressed "Apply" and the feedback was given to the classifier. An *Undo* button at the bottom of the feedback panel allowed participants to undo their previous action.

Besides the above communications about keywords, participants could also communicate about folders. When the folder displayed in the *folder* column of the inbox was correct and the user did not wish to do any further manipulations, s/he could "file it" (by pressing the *File It* button). This moved the email to the predicted folder (or, the participant could select a different folder from the dropdown menu in the feedback panel).

Once changes were applied, the program updated the predicted folders of the emails in the inbox. Emails for which the classifier changed the predicted folder after user feedback were highlighted in red in the email list.

Whenever an email was filed away, the system provided information about the potential benefits and risks of the participant's changes in the status panel (middle left). These were to motivate the participant, and to provide a reasonably accurate assessment on the progress (or harm) their changes were causing.

Classification Algorithm

A key question in incorporating user feedback into a machine learning algorithm is how to do the actual incorporation. One general approach is to treat user feedback as hard constraints to the algorithm. For instance, the constraints can enforce qualitative monotonicities [1], clamp labels in a Conditional Random Field (CRF) [11], fix parameters in a graph model [19] or incorporate prior knowledge into support vector machines [15]. Other work uses the feedback to select features for the learning algorithm [21, 14]. In [29], the authors let the user directly build a decision tree for the data set with the help of visualization techniques.

In our preliminary work [28], we incorporated user feedback into a Naïve Bayes classifier by converting the feedback into a set of constraints. During training, the parameters of the classifier were calculated through a constrained optimization procedure which maximized the likelihood of the data given the constraints provided by the user feedback. Unfortunately, this approach either decreased classification accuracy or produced little improvement. In some cases, the constraints were already satisfied, leading to no changes to the classifier's behavior. In other cases, the feedback over-constrained the learning algorithm, resulting in sub-optimal settings of the classifier's parameters.

As a result, we have been exploring an approach called *user co-training*, which leverages user feedback more aggressively. Because our preliminary results have been encouraging, particularly in situations in which training data was scarce, we selected that approach for the current experiment.

User co-training is similar to the co-training algorithm [5] used in semi-supervised learning. Semi-supervised learning [8] is used when labeled data is limited but unlabeled data is abundant; its goal is to improve the performance of a learning algorithm trained on the small amount of labeled data by leveraging the structure of the unlabeled data. Co-training employs two classifiers that work on the same data but have two different "views" of the data through independent sets of features. The two classifiers are assumed to produce the same classification even though they have different views. Initially, the two classifiers are trained on a labeled training set. Then, in the second phase of training, the classifiers compare which of the two can more confidently classify a training instance from the unlabeled data. The most confidently classified training instance is labeled and added to the training set for the next round of training.

We adapted the notion of co-training by regarding the *user* as one of the classifiers in co-training and using this classifier to label data for a second classifier, which was a Naïve Bayes algorithm in our study. In order to treat the user as a classifier, we developed a *user feedback classifier* that represents the user and treats the keywords selected in the user feedback as a set of features for the specific folder to which the user assigns the email. In order for a keyword to be selected by the user, the user must either have added the keyword or modified the weight of the keyword. If a keyword is deleted by the user, that keyword is removed from the set of features used by the classifier. Thus, associated with each folder f is a vector of keywords $\boldsymbol{v}_f$ obtained by taking the union of all the user-selected keywords in the email messages placed into folder f. The weight of each component in the vector $\boldsymbol{v}_f$ is determined by the votes proposed by the user in the feedback[2]. The weight of each type of vote ranges from 0 to 2.0 as shown in Table 1, with the exact amount determined by the position of the vote slider bar set by the participant.

Vote	Weight
Do Not Change	0
Increase a tiny bit	0.01-0.39
Increase a bit	0.40-0.79
Increase some	0.80-1.19
Increase a lot	1.20-1.59
Increase a lot!!	1.60-2.00

Table 1. Weights for each vote.

Let F be the set of all folders.
For each folder f, create a vector $\boldsymbol{v}_f$ with the voted weights of the user-selected keywords
For each message m in the unlabeled data

$$\boldsymbol{FolderScore}_f = \text{sum of weights in } \boldsymbol{v}_f \text{ of keywords appearing in } m$$

$$\boldsymbol{f}_{max} = \arg\max_{f \in F} \boldsymbol{FolderScore}_f$$

$$\boldsymbol{FolderScore}_{other} = \max_{f \in F \setminus f_{\max}} \boldsymbol{FolderScore}_f$$

$$\boldsymbol{Score}_{\mathbf{m}} = \boldsymbol{FolderScore}_{fmax} - \boldsymbol{FolderScore}_{other}$$

Sort $\boldsymbol{Score}_{\mathbf{m}}$ for all messages in decreasing order
Select the top $\boldsymbol{k}$ messages to add to the training set along with their folder label $\boldsymbol{f}_{max}$

Figure 2. Our user co-training algorithm.

Just as in standard co-training, we iteratively increment our training set with the emails that are most confidently as-

[2] Even though the weight modification is called a "vote", the effects are not cumulative. Each component in the vector $\boldsymbol{v}_f$ is set to the vote weight. For instance, if the user selects the keyword "bankrupt" and votes "Increase a lot!!" twice with a value of 2.0, the weight on the keyword bankrupt will be set to 2.0 and not 4.0.

signed to folders by the user feedback classifier[3]. In order to determine our confidence in the folder assignment, we derive a score for each email message. This score is indicative of the gap between the best folder as predicted by the weighted user-selected keywords, and the next best predicted folder. The emails with the top *k* scores along with their folder predictions are then added to the training set for the Naïve Bayes classifier. Pseudocode for the user co-training algorithm is shown in Figure 2. We set the value of *k* to be 10 as there was little change in the results for values of $k > 10$.

RESULTS

How Much Did Feedback Improve the System?

Classifying email accurately by machine learning is a very challenging problem, as has already been reported by several supervised learning efforts to automatically classify email messages into categories defined by users [6, 9, 26]. The challenges stem from numerous factors, such as imbalanced categories, incomplete information in the email messages, and the fact that the categories (folders) set up by the users are often idiosyncratic and non-orthogonal.

What about accuracy in classifying email by humans? In order to assess participant-introduced error in our experiment we calculated their *filing accuracy* as the accuracy of folder assignment made by participants compared to the folder assignment made by the original Enron users (Table 2, second column). (For folder assignment, the participant either filed the email in a folder or applied a corrected folder from the drop-down menu.) In our experiment, the average filing accuracy was 64.2%.

This error rate is high. It is possible that the amount of user-introduced error could have been reduced if participants could have worked on their own emails, but in other human data relating to judgments and classifications (e.g., [25]), error rates up to 20% have been reported. Clearly, human inconsistencies and errors in classifying email are a threat in this domain.

This fact underscores the importance of machine learning algorithms in this domain dealing with some amount of error. Accuracy improvements are particularly challenging if machine learning needs to guard against substantial error rates while at the same time responding to user feedback.

Turning to machine accuracy, in order to compare the effects of user feedback on the machine learning algorithm, we created three versions of the email classifier, each trained in ways appropriate to each version. Specifically:

Baseline: The user co-training algorithm was trained using the 50 emails in the original training set along with the emails in the feedback set that participants had assigned to folders. (Once again, for folder assignment, the participant either filed the email in a folder or applied a corrected folder from the drop-down menu.) For the emails from the feedback set, we trained the classifier using the *actual* folder assignments made by the Enron users as the class label.

Folder Feedback: Similarly, we trained this classifier using emails from the original training set along with emails in the feedback set that participants had assigned to folders. However, for the emails from the feedback set, we trained the Folder Feedback classifier using the *participant's* folder assignments instead of the Enron users' folder assignments.

Rich Feedback: The user co-training algorithm was trained

Participant	Participants' Filing Accuracy	Baseline	Folder Feedback	Rich Feedback
101	0.700	0.544	0.321	0.516
105	0.429	0.096	0.286	0.100
109	0.667	0.106	0.102	0.553
110	0.560	0.105	0.300	0.649
111	0.794	0.438	0.247	0.257
202	0.757	0.317	0.108	0.109
204	0.564	0.103	0.111	0.106
206	0.340	0.105	0.502	0.500
207	0.658	0.293	0.097	0.287
208	0.760	0.469	0.108	0.114
213	0.694	0.547	0.320	0.500
214	0.588	0.098	0.094	0.101
215	0.313	0.287	0.272	0.304
217	0.720	0.113	0.096	0.120
301	0.844	0.496	0.575	0.662
303	0.767	0.493	0.493	0.768
304	0.560	0.177	0.327	0.297
305	0.723	0.199	0.120	0.288
312	0.760	0.495	0.578	0.106
316	0.440	0.498	0.327	0.310
3001	0.280	0.116	0.500	0.109
3003	0.689	0.172	0.290	0.284
3004	0.923	0.559	0.503	0.101
3005	0.320	0.104	0.102	0.104
3006	0.660	0.109	0.148	0.163
3008	0.740	0.500	0.205	0.157
3009	0.711	0.112	0.153	0.297
3102	0.718	0.314	0.283	0.287
3103	0.667	0.329	0.391	0.381
3105	0.923	0.350	0.505	0.499

Table 2. The accuracy of the Baseline, Folder Feedback, Rich Feedback classifiers on the independent test set.

[3] This variant of user co-training is slightly different from the version used in [28] where we multiplied the sum of the components in $\boldsymbol{v}_f$ with the posterior probability of the most likely folder as calculated by the Naïve Bayes classifier.

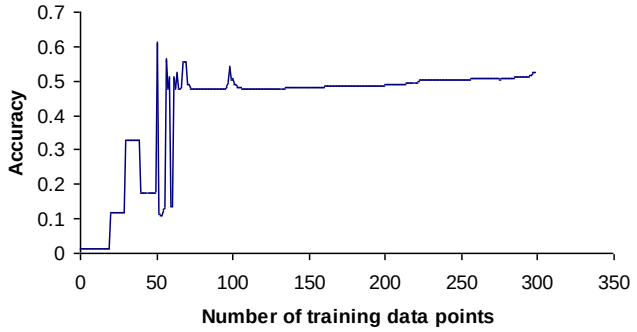

Figure 3. A plot of accuracy versus the number of training data points for Participant 101.

using the original training set plus all forms of user feedback in the feedback set, i.e. both folder assignments and keyword modifications.

We then evaluated these versions on the test set (described in the experiment set-up section). The results, shown in Table 2, indicate that the Rich Feedback classifier improved accuracy over the Baseline classifier for 60% of the participants. In addition, the Rich Feedback classifier outperformed the Folder Feedback in 60% of the cases (for a different group of participants from those in the Baseline comparison). Rich Feedback sometimes decreased accuracy, including decreases of 46% (vs. Baseline) and 47% (vs. Folder Feedback) in the worst cases. The cases in which it increased accuracy were similarly dramatic, with the best improvement at 54% (vs. Baseline) and 45% (vs. Folder Feedback).

What led to improvements in Rich Feedback accuracy? We found that participants with a high average or maximum time between filing emails had a higher Rich Feedback accuracy (linear regression, p=0.0466 and p=0.0240, respectively). Likewise, participants with more changes to folder assignments had a higher accuracy (linear regression, p=0.0227). Changing folder assignments allowed participants to see the keywords associated with other folders. These findings suggest that participants who took more care in creating their rich feedback were more effective in improving the classifier's accuracy.

For a close-up view of accuracy patterns, consider Figure 3, which shows the Baseline classifier accuracy for participant 101 over a test set of 800 email messages as a function of the number of training examples. To gain insights into the behavior of our classifier, we added more training examples to the original training set of size 50 by moving email messages from the test set to the training set[4]. All training examples had the original Enron users' folder assignments as the class label.

One would typically expect to see a steady improvement in classification accuracy. However, this improvement only happened after about 70 training examples. The accuracy fluctuated wildly between 0-70 training examples, with a dramatic drop in accuracy at the 51st training example. This unstable period is due to the classifier not having enough training data and having its classification boundaries dramatically changed each time it saw a new training example. Further analysis at training example 50 showed that the classifier classified many emails from the Systems folder correctly, but at training example 51, almost none of the emails from the Systems folder were classified correctly. This behavior occurred in all the participants' classifiers during the unstable period.

These results have implications for the design of machine learning algorithms that can incorporate rich feedback. We had initially hypothesized that user feedback would be most helpful during the initial stages when training data is limited, but our results point out the importance of being wary of the initial unstable periods, which can be very frustrating for users, as we will discuss in the next section. Early start-up periods aside, there is little research to date into what types of machine learning algorithms can be used effectively in an interactive setting where a user can modify the learning algorithm directly. We have investigated a user co-training approach in this work as well as a constraint-based approach in our preliminary work. However, many other learning algorithms remain unexplored.

The Rich Feedback System from the Users' Perspective

Users' willingness and effectiveness at interacting with the system are an essential part of intelligent user interfaces. To investigate how users might perceive the opportunities for rich communication, we analyzed the post-session questionnaires as to participants' ratings and reasons for these ratings.

Participants in general rated the system as acceptable, but there is room for improvement. Responses to TLX questions (Mental Demand, Temporal Demand, Success of Performance, Effort, Frustration) were all around the mid-point of the 5-point Likert scale. Participants also gave neutral ratings for ease of feedback (mean = 3.07, std.dev. = 1.12), trust if they could verify the correctness of the predictions, and whether they would recommend the program to a friend (mean = 2.86, std.dev. = 1.05). There appeared to be no overall problems with understanding how the suggestions worked (mean = 3.21, std.dev. = 1.01).

What Participants Said

Where were the opportunities for improvement in the users' perspectives? To consider this question, we analyzed the comments that participants gave explaining their ratings.

We used an affinity diagramming process [18] to develop codes bottom-up from the questionnaire comments. This

[4] The step-function behavior before 50 training data points is due to the fact that the original training set of size 50 contained the emails sorted by folder. The additional emails added to the original training set were randomly chosen from the test set.

process allows large amounts of items to be organized into high-level concepts that have high consensus within a group of researchers. We proceeded as follows: Each comment was divided into thematic statements. Each statement was assigned randomly to one of three researchers. Each researcher sorted statements individually into groups and wrote down higher-level concepts that described the groups' content. Researchers could use any existing groups in addition to their own. After all statements had been sorted, the concepts were reviewed by all researchers and supercodes were generated if necessary. We validated the reliability of the codes through a consistency check. Two researchers independently coded the statements. They then further refined the codes and developed norms about how to apply them. For the coding scheme, the total agreement value was 79%, which indicates high coding reliability.

Supercodes were used when responses differed in some particular aspect of content but were related in a general high-level concept. For example, some participants made general comments on how the system was learning but other participants commented more specifically on the system's learning changing too much or too little. As a result, the "Wall of red" and "System learning too little" codes are part of the supercode of "Effects of system learning".

We calculated the frequency of these codes based on the number of participants that gave these respective responses. A participant could make a comment in a response to a question that covered more than one code. If a code was found to be repeated in comments to several questions we only counted it once. Subcodes also counted as an occurrence of the supercode. Table 3 shows the nine codes that resulted from this process.

System Learning and User Control: Heed Me!

The supercode "Effects of System Learning" was the most common, occurring in 28 participants (65%). Of these, 12 participants (28%) commented that the system was learning too much, and 16 (37%) commented that it was learning too little.

A general theme was that participants were willing and perhaps even eager to provide direction to the system, but when they did so, they expected better obedience from the system for the amount of effort they expended. A higher rating for "frustration" and a lower rating for "ease of feedback" were predictive for mentioning these kinds of problems (logistic regression, p=0.00448 and p=0.00456, respectively). There was also a predictive relationship between the number of emails filed and rating for overall effort (linear regression, p=0.04391, R^2=0.1372, F[1,28]=4.453).

GPA scores were also predictive of frustration ratings (linear regression, p=0.01162, R^2=0.1455, F[1,41]=6.979). Academic success is sometimes attributed to critical reasoning [13] and it may be that participants felt particularly frustrated if they understood how to critique the system but their changes were not heeded closely enough. For example:

> *P3102: "I conceptually understood the basic structure - higher frequency of more weighted words led to emails placed in category A vs. category B - but what I did frequently seemed to have little effect."*

A practical implication of these reactions is that, if a machine learning system provides the ability for users to offer suggestions for reasoning changes, the users expect not only that those suggestions be heeded, but also that they be able to *detect* the fact that their suggestions are being heeded.

Granularity of User End of the Dialog: Folders or Words?

Eighteen participants (42%) expressed confusion or diffi-

Code	Frequency	Example
Keywords	31 (72%)	"You had to be careful on what keywords to assign because many of them could show up anywhere."
Effects of system learning	28 (65%)	"I understood what to do, but it was difficult at times to get the system to do what I meant for it to do."
Folders	18 (42%)	"If the program flagged emails that weren't strongly placed in a file or emails that could fit easily into 2 or more folders"
Effects of system learning : System learning too little	16 (37%)	"Some of the commands I gave to the computer were not performed by the computer."
Communication of system changes	16 (37%)	"The system kept going back saying the emails filed will now be going to a different folder after I changed or deleted a keyword."
Transparency about the system's internal workings	15 (35%)	"Not satisfied, would like to know more about how it works or at least the mechanics behind it."
Effects of system learning/ *Communication of system changes*: Wall of red	12 (28%)	"It seemed like every time I changed one keyword by just a little, ALL the emails would suddenly switch into that folder."
Communication of system changes: Communications from the status panel	8 (19%)	"I couldn't get the system to file at even a 50% success rate."
Unlearn	7 (16%)	"The ability to later move something to a different folder if it ended up in the wrong one."

Table 3. Codes and their frequency of occurrence in participants' comments

culties using folder assignments as a means to provide feedback. A frequent suggestion was that the system should be responsive to moving emails between folders, especially if participants changed their minds. Some participants reported that it was unclear that keywords were specific to folders. Problems were also caused by only being allowed to assign one folder as the correct folder.

Participants commented more heavily on the use of keywords as a basis for learning (31 participants, 72%). Many of them were concerned with the difficulty of finding words that could represent one folder well and not appear in other folders:

> *P3102: "Language is ridiculously malleable, and so choosing the words that would only apply in a given scenario, or would apply significantly, was difficult."*

As a result, participants who commented on keywords often wanted more methods of using keywords for feedback besides the "add keyword", "remove keyword" and "increase weight" they were given in the program. Common suggestions included the ability to decrease the weight of words, to add rules based on keywords, to specify context of keywords in the structure of an email (specifically send and receive fields), and the use of phrases:

> *P0208: "The program doesn't allow for groups of words to be considered as one. Such as "love you baby" or other obvious phrases "I'll get back to you" or "on my desk" appear often and would make the program more effective."*

Our findings have two practical implications. First, users should be able to provide folder assignments more flexibly, especially if folder organization is not orthogonal. Second, more refinements to the feedback mechanisms are needed, and feedback mechanisms need to be extended to cover keywords combinations and parsing/extracting keywords in a different way, relational features, and even wholesale changes to the algorithm.

The System's End of the Dialog

Communication from the system was important to many participants. Sixteen participants (37%) made comments of type "Communication of System Changes". The reasons can be attributed to two different communication mechanisms we employed in the program.

First, emails where folders had changed due to learning were highlighted in red. Twelve participants (28%) reacted strongly to this. At issue was the fact that, especially in the early stages of adding feedback when the classifier was unstable, feedback could change most or all the messages, creating a "wall of red":

> *P0313: "A small change swung most of the inbox emails to a given folder."*

> *P3004: "You had to be careful on what keywords to assign because many of them could show up anywhere…It seems that by changing [the weight of] one word could potentially change all email classifications at once. It was a little bit like 1 step forward, 2 steps back at times."*

Second, there was the status panel. Eight participants (19%) mentioned this panel in the post-session questionnaire, correctly interpreting the information as progress on their feedback:

> *P3102: "At the end, I was only around 50% successful at showing the program which emails could be filed where I wanted them."*

Many participants found communications by the system to be inadequate. Fifteen participants (35%) had comments in the "Transparency" category, reflecting the need for better explanation or understanding of how the system worked. Participants' suggestions for how to improve the system's transparency varied, but most involved making some change in how keywords were presented (80% of participants who mentioned the "Transparency" code also mentioned the "Keyword" code). A common request was for the system to provide a "master list" of keywords for each folder:

> *P3003: "I would like to be able to view a master list of all the keywords for each folder and their importance."*

A practical implication for machine learning systems that take rich feedback into account is that explanations are needed that allow the user to choose *good* feedback. In particular, users need to see what influence keywords have on prediction choices.

Types and Timing of Feedback

Appropriate Feedback

In our previous study [27] keyword changes covered the majority of rich feedback that participants gave (53% different feature selection, 12% weight adjustment). Therefore, in this experiment, we provided mechanisms for participants to add keywords, delete keywords and change the weights on keywords.

We found that, in aggregate, participants used these feedback mechanisms about equally over the course of the experiment, although they tended to display individual preferences for certain feedback mechanisms. Table 4 shows the average amount that each feedback mechanism was used by participants. When participants added a keyword, they frequently also adjusted the weight of that keyword at the same time.

There was some evidence that participants carefully consid-

	Mean	Std. Dev.
Add keyword	37.6	29.2
Delete keyword	37.8	56.2
Weight Change	40.4	28.9

Table 4. Mean Keyword Changes Made by Participants

ered their feedback choices. Participants altered 9% of their additions, deletions and weight changes to keywords and 75.2% of their folder assignments before they committed them to the learning system, suggesting a fair amount of experimentation before settling upon their desired keyword manipulations.

These findings suggest that these mechanisms for allowing rich feedback are viable ways of engaging users in the reasoning process. Selecting different keywords and making changes to weights were all used by our participants with great care, and lend themselves for straightforward integration into intelligent user interfaces.

Unlearning

Current machine learning algorithms do not allow "unlearning" directly. Instead, the only way that a concept can be unlearned is to provide enough new training examples that contradict previous training data or if new data is weighted more heavily. In our experiment we provided an Undo button which allowed the participants to retract any feedback that they had made to the learning system and revert to the previous state of the classifier. This made the system explicitly unlearn feedback. Participants used this approach for 1.32% of their changes to keywords. Seven participants (16%) also mentioned the need to unlearn in the post-session questionnaire.

System communication plays a key role in identifying undesired changes that may make the system less accurate, and when the system should unlearn. We found that participants were more likely to undo changes if they resulted in many new changes in the inbox *and* the newly predicted folders of those emails were incorrect (logistic regression, p=0.00052).

There are several implications for the design of machine learning systems. Even simple communications from the system about its reasoning can be effective ways to allow the user to assess the changes made and which changes were undesirable.

Our findings also indicate that other ways to tell the system to unlearn, in addition to giving more counter-examples, are important to users. The ability to unlearn a concept is an aspect that has received little attention in machine learning systems.

Gender Differences

Gender may influence how feedback to learning is given. Recent research has reported gender differences in males' versus females' interest in exploring and experimenting with innovative features [2]. In our study, females took on average 6.64 minutes longer than males to complete the experiment (two-sample t-test, p=0.0279), but there was no significant difference in the number of emails filed for males and females. In our experiment, we found that females added nearly twice as many keywords (mean=49.7) as males did (mean=25.5) (two-sample t-test, p=0.0208). Although females changed more keyword weights (mean=49.8) than males (mean=31), this difference was not significant, and there was no difference in number of keywords deleted.

The importance of guarding against undesired changes may also be influenced by gender. Six out of the seven participants who commented on the need for the system to be able to unlearn in the post-session questionnaire were female. This is consistent with previous research showing that females perceive more risk than males in tasks involving mathematical or spatial reasoning [7].

IMPLICATIONS AND CONCLUSION

Our results have practical implications for the design of intelligent user interfaces that take rich feedback into account. First, from a user perspective, we have found that appropriate feedback mechanisms and system communication played important roles. We found that participants placed emphasis on being heeded by the system if they gave feedback. We showed that participants used feedback mechanisms with care to guard against undesired effects. The choice of feedback mattered; participants strived for good keywords and requested a greater variety of mechanisms to allow better feedback. The communication from the system mattered here too. Participants expressed the need for explanations that helped them in their choice of feedback.

From a machine learning standpoint, we have shown empirical results of incorporating rich user feedback into classification. We showed that user feedback, employing the mechanisms we devised, can improve accuracy considerably, especially if a machine learning system must be responsive to few initial training examples or to changes in classification. We have discovered wild swings in decision boundaries during early, unstable stages of classifier training. This can frustrate users, and algorithms accepting rich user feedback must alleviate this problem.

We have also identified open research questions. The ability to unlearn learning was important to our participants, and this has not been addressed in machine learning systems. Gender may have played a role in how feedback mechanisms were used and in unlearning learning; there has been little research about gender in relation to intelligent user interfaces. In future work, we would like to explore more sophisticated forms of rich user feedback as well as develop new machine learning algorithms for responding to this feedback.

Our experiment provided some positive initial results yet also underlined the challenges that incorporating rich user feedback poses. This suggests more steps in exploring ways in which intelligent user interfaces can incorporate the intelligence of users.

ACKNOWLEDGEMENTS

We thank the participants of our study. We also thank Lida

Li for his assistance. This project was supported in part by NSF (grants IIS-0133994, CCF-0325273, ITWF-0420533), by Intel, and by DARPA (grant HR0011-04-1-0005, contract NBCHD030010).

REFERENCES

1. Altendorf, E., Restificar, E., Dietterich, T. Learning from sparse data by exploiting monotonicity constraints. *Proc. UAI* (2005).
2. Beckwith, L. Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., Cook, C. Tinkering and gender in end-user programmers' debugging, *Proc. CHI 2006*, ACM Press (2006), 231-240.
3. Billsus, D., Hilbert, D., Maynes-Aminzade, D. Improving proactive information systems. *Proc. IUI* (2005), 159-166.
4. Blythe, J. Task learning by instruction in Tailor. *Proc. IUI* (2005), 191-198.
5. Blum, A., Mitchell, T. Combining labeled and unlabeled data with co-training. *Proc. COLT* (1998).
6. Brutlag, J., Meek, C. Challenges of the email domain for text classification. *Proc. ICML* (2000), 103-110.
7. Byrnes, J. P., Miller, D. C., Schafer W. D. Gender differences in risk taking: A meta-analysis. *Psychological Bulletin* 125 (1999), 367-383.
8. Chapelle, O., Scholkopf, B., Zien, A. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
9. Cohen, W. Learning rules that classify e-mail. *Proc. AAAI Spring Symp. Information Access* (1996).
10. Crawford, E., Kay, J., McCreath, E. IEMS – The Intelligent Email Sorter. *Proc. ICML* (2002), 83-90.
11. Culotta, A. Kristjansson, T. McCallum, A., Viola, P. Corrective Feedback and Persistent Learning for Information Extraction. *Artificial Intelligence 170*, (2006), 1101-1122.
12. Cypher, A. (ed.) *Watch What I Do: Programming by Demonstration*, MIT Press, Cambridge, MA, 1993.
13. Facione, P. A. The California Critical Thinking Skills Test: College Level Technical Report #2. California Academic Press, Millbrae CA. 1990 (ERIC Document Reproduction Service No. ED 327 550).
14. Fails, J. A., Olsen, D. R. Interactive machine learning. *Proc. IUI* (2003), 39-45.
15. Fung, G., Mangasarian, O., Shavlik, J. Knowledge-based support vector machine classifiers. *Proc. NIPS* (2002).
16. Hart, S., Staveland, L. Development of a NASA-TLX (Task load index): Results of empirical and theoretical research, *Human Mental Workload* (1988), Hancock, P. and Meshkati, N. (eds.), 139-183.
17. Herlocker, J., Konstan, J., Riedl, J. Explaining collaborative filtering recommendations. *Proc. CSCW* (2000), 241-250.
18. Holtzblatt, K., Beyer, H. Making customer-centered design work for teams. *Comms ACM 36*, 10 (1993), 92-103.
19. Huang, Y., Mitchell, T. M. Text clustering with extended user feedback. *Proc. SIGIR* (2006), 413-420.
20. Lieberman, H., (ed.) *Your Wish is My Command: Programming By Example*. 2001.
21. Liu, B. Li, X. Lee, W., Yu, P. Text Classification by Labeling Words. *Proc. AAAI* (2004).
22. McCarthy, K., Reilly, J., McGinty, L., Smyth, B. Experiments in dynamic critiquing. *Proc. IUI* (2005), 175-182.
23. McDaniel, R.G. and Myers, B.A. Getting more out of programming-by-demonstration. *Proc. CHI* (1999), 442-449.
24. Oblinger, D., Castelli, V., Bergman, L. Augmentation-based learning. *Proc. IUI* (2006), 202-209.
25. Phalgune, A., Kissinger, C., Burnett, M., Cook, C., Beckwith, L. Ruthruff, J. Garbage in, garbage out? An empirical look at oracle mistakes by end-user programmers. *Proc. VL/HCC* (2005), 45-52.
26. Shen, J., Li, L., Dietterich, T. Herlocker, J. A hybrid learning system for recognizing user tasks from desk activities and email messages. *Proc. IUI* (2006), 86-92.
27. Stumpf S, Rajaram V, Li L, Burnett M, Dietterich T, Sullivan E, Drummond R, Herlocker J. Toward Harnessing User Feedback For Machine Learning. *Proc. IUI* (2007).
28. Stumpf S, Rajaram V, Li L, Wong W-K, Burnett M, Dietterich T, Sullivan E, Herlocker J. Interacting Meaningfully with Machine Learning Systems: Three Experiments. EECS OSU Technical Report 2007-46, http://eecs.oregonstate.edu/library.
29. Ware, M., Frank, E., Holmes, G., Hall, M., Witten, I. H. Interactive machine learning: letting users build classifiers. *IJHCS 55* (2001), 281-292.