Interacting Meaningfully with Machine Learning Systems: Three Experiments[1]

Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong,
Margaret Burnett, Thomas Dietterich, Erin Sullivan, Jonathan Herlocker

Oregon State University
School of Electrical Engineering and Computer Science
Corvallis, OR 97331 USA
1-541-737-3617
{stumpf,rajaramv,lili,wong,burnett,tgd,sullivae,herlock}@eecs.oregonstate.edu

ABSTRACT

Although machine learning is becoming commonly used in today's software, there has been little research into how end users might interact with machine learning systems, beyond communicating simple "right/wrong" judgments. If the users themselves could work hand-in-hand with machine learning systems, the users' understanding and trust of the system could improve and the accuracy of learning systems could be improved as well. We conducted three experiments to understand the potential for rich interactions between users and machine learning systems. The first experiment was a think-aloud study that investigated users' willingness to interact with machine learning reasoning, and what kinds of feedback users might give to machine learning systems. We then investigated the viability of introducing such feedback into machine learning systems, specifically, how to incorporate some of these types of user feedback into machine learning systems, and what their impact was on the accuracy of the system. Taken together, the results of our experiments show that supporting rich interactions between users and machine learning systems is feasible for both user and machine. This shows the potential of rich human-computer collaboration via on-the-spot interactions as a promising direction for machine learning systems and users to collaboratively share intelligence.

1. INTRODUCTION

A new style of human-computer interaction is emerging, in which some reasoning and intelligence reside in the computer itself. These intelligent systems and user interfaces attempt to adapt to their users' needs, to incorporate knowledge of the individual's preferences, and to assist in making appropriate decisions based on the user's data history. These approaches use artificial intelligence to support the system's part of the reasoning. One increasingly common approach being brought to intelligent systems and 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 the 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.

Although machine learning is often reasonably reliable, it is rarely completely correct. One factor is that statistical methods require many training instances before they can reliably learn user behavior. Sometimes correctness is not critical. For example, a spam filter that successfully collects 90% of dangerous, virus-infested spam leaves the user in a far better situation than having no spam filter at all. But sometimes overall correctness or even correctness for certain types of data (such as data from a specific class or data with a particular combination of features) is important. For example, recommender systems that recommend substandard suppliers or incorrect parts, language translators that translate incorrectly, decision support systems that lead the user to overlook important factors, and even email classifier algorithms that misfile important messages could cause significant losses to their users and raise significant liability issues for businesses. Further, too much inaccuracy in "intelligent" systems erodes users' trust.

[1] An early presentation of Experiment #1 appeared in Proceedings of ACM IUI'07 (Stumpf et al. 2007).

When accuracy matters, allowing the user to help could make a crucial difference. Therefore, approaches have begun to emerge in which the user and the system *interact with each other*, not just to accomplish the goal of the moment, but also *to improve the system's accuracy* in its services to the user over the longer term. Our work falls under the general category of mixed initiative user interfaces, in which users collaboratively interact with intelligent systems to achieve a goal (Horvitz 1999).

As Section 2 explains, this direction is still in its infancy: the norm for the few machine learning systems that communicate with users at all 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 how to make the correct prediction. We began to consider whether end users might be able to provide rich guidance to machine learning systems. We wondered whether enabling them to provide this guidance could substantially improve the speed and accuracy of these systems, especially early on in training when machine learning does not possess a lot of knowledge.

Many questions arise from this possibility. Will users be interested in providing rich feedback to machine learning systems? If so, what kind of advice will they give? Will their feedback be usable by machine learning algorithms? If so, *how* would the algorithms need to be changed so that they could make use of this feedback? In this paper, we begin an exploration of these questions. The overall premise we explore is that, if the machine learning system could explain its reasoning more fully to the user, perhaps the user would, in return, specify *why* the prediction was wrong and provide other, rich forms of feedback that could improve the accuracy of machine learning.

There are implications for both directions of the communication involved in this premise. First, the system's explanations of why it has made a prediction must be usable and useful to the user. Second, the user's explanations of what was wrong (or right) about the system's reasoning must be usable and useful to the system. Both directions of communication must be viable for production/processing by both the system and the user.

As a first step to investigate possibilities for both directions of communication, we conducted three studies. The first was a think-aloud study with email users, to inform the design of future machine learning systems aiming to support such communications. In this first study, machine learning algorithms sorted email messages into folders and explained their reasoning using three different explanation paradigms: Rule-based, Keyword-based, and Similarity-based. The participants were asked to provide feedback to improve the predictions. No restrictions were placed upon the form or content of participants' feedback. We then conducted two follow-up experiments, in which we changed our machine learning algorithms to make use of some of what the users advised in the first study, evaluating the results. In both of these experiments, we conducted off-line experiments to investigate *how* some of the user feedback could be incorporated into machine learning methods and to evaluate the *effectiveness* of doing so.

Thus, our research questions were:

RQ 1. How can machine learning systems explain themselves such that (a) end users can understand the system's reasoning, and (b) end users are willing to provide the system rich, informative feedback with potential to improve the system's accuracy?

RQ 2. What types of feedback will end users give? That is, how might we categorize the nature of their feedback from the perspective of machine learning, and what sources of background knowledge underlie users' feedback?

RQ 3. Can these types of user feedback be assimilated by existing learning algorithms? If so, exactly how could some of these types of user feedback be incorporated into machine learning algorithms, and does doing so actually improve the performance of algorithms?

2. RELATED WORK

2.1 Explanations

Any effort to obtain user feedback about a learning system needs to ensure that explanations by the learning system are understandable and useful to users. Previous work has shown that explanations of system behavior are extremely beneficial. Explanations that answer why certain outcomes happened, based on user actions, can contribute positively to system use (Herlocker et al. 2000; Myers et al. 2006; Crawford et al. 2002a, 2002b). Similarly, it has been shown that highlighting the relationship between user actions and ensuing predictions can influence user preference (Billsus et al. 2005). There is also previous research on the characteristics of explanations that help users choose between predictions. For example, showing contrasting features in recommendations can play a role in user trust (Herlocker et al. 2000; Pu and Chen 2006). One way that this relationship can be expressed is by making use of various ways of reasoning, such as analogical reasoning (Lieberman and Kumar 2005). There is some evidence that transparency via explanations is a major factor towards establishing trust in adaptive agents (Glass et al. 2008; Pazzani 2000).

As a first step the learning system itself needs to be capable of producing a comprehensible explanation of its results. The bulk of the literature on explaining intelligent systems has focused on explaining expert systems based on deductive logic or rules (Swartout 1983; Clancey 1983; Wick and Thompson 1992; Johnson 1994). In contrast, much less attention has been given to explaining learning systems. Although some learning algorithms such as decision trees (Quinlan 1993) are capable of readily providing explanations of the underlying reasoning, the majority of learning algorithms employ a statistical or mathematical approach, which is much more challenging to explain than a logic-based system. Prior work has focused on linear additive models that associate a weight with each feature and produce a result by summing up these weighted features. These linear additive models, such as naïve Bayes, logistic regression and linear support vector machines, can be explained through showing how the weight on each feature contributes to the overall prediction and a graph or chart can be used for visualization (Becker et al. 2001; Kononenko 1993; Mozina et al. 2004; Poulin et al. 2006, Ridgeway et al. 1998). Apart from linear additive classifiers, explanation methods have also been developed to explain the computation of posterior probabilities for Bayesian networks. These computationally expensive explanation algorithms for Bayesian networks consist of finding the subset of evidence that most influences the posterior probability and visualizing the flow of evidence through the network during inference (Suermondt 1992; Chajewska and Draper 1998; Lacave and Diez 2002).

These existing explanation techniques are helpful to machine learning experts who understand the underlying algorithms but they are typically not reasonable for an end user with no background in machine learning. Another explanation strategy is to create a more comprehensible representation of the learned model. This approach balances the issues of fidelity and comprehensibility (Craven 1996; vand de Merckt and Decaestecker 1995). Examples of these approaches include pruning to simplify decision trees (Bohanec and Bratko 1994; Cunningham et al. 1996) and inducing comprehensible models, such as decision trees (Craven 1996), from neural networks, which are notoriously difficult for a human to understand. These approaches to explanation are purely intended to illustrate the system's behavior to the end user and abstract away from the actual details of the underlying algorithm. All of the explanation techniques mentioned thus far are static presentations to the end user. In our approach, we require the explanations to be interactive such that changes to the explanations made by the user can be incorporated back into the machine learning algorithm.

2.2 Incorporating User Feedback

We also require end users to provide corrective feedback to these explanations and this feedback needs to be incorporated into the machine learning algorithm. Different methods for gathering user feedback have been investigated, along a spectrum of formality and richness. An obvious way to gather user feedback is to allow interactions in natural language (Blythe 2005). Semi-formal types of feedback that have been shown to be preferred by users make use of editing feature-value pairs (McCarthy et al. 2005; Chklovski et al. 2005). 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 (Cypher 1993; McDaniel and Myers 1999; Lieberman 2001). For example, the process might be ordering an item from an on-line web-site. With some exceptions (Oblinger et al. 2006), the user is not usually given the opportunity to provide feedback about the reasoning behind the learned program. So far there has been a lack of research that integrates an investigation into the understanding of machine learning systems' explanations with an analysis of the *content* of the rich feedback users give when they have an unconstrained opportunity to do so.

Once the user feedback has been obtained, it needs to be incorporated into the machine learning algorithm. One of the simplest techniques for incorporating a user's feedback is to allow the user to interactively generate additional training examples. This technique is employed in the CRAYONS system (Fails and Olsen 2003), in which the user interactively selects foreground and background pixels in an image. In machine learning, one of the most common approaches for interacting with an end user is active learning (Mackay 1992; Cohn et al. 1996; Tong and Koller 2002). An active learning algorithm operates in a loop in which the first step is to ask the user to label a training instance that would provide the most information regarding the classification task. Once the label is provided by the user, the active learning algorithm updates its parameters with the new training instance and the loop continues if the accuracy of the classifier is still unsatisfactory. All of the approaches mentioned thus far interact with the user through labels on the training examples. While labels are typically easy for a user to provide, the user may need to provide a large number of labels before the desired level of accuracy is reached. A potentially more efficient way to incorporate user feedback is to provide feedback that extends beyond just providing class labels. This feedback, which we will call *rich feedback*, involves incorporating more expressive forms of corrective feedback which can cause a deeper change in the underlying machine learning algorithm.

Incorporating rich feedback into a machine learning algorithm is a special case of incorporating domain knowledge into a learning algorithm. There is a vast body of literature on incorporating domain knowledge into learning (eg. Towell and Shavlik 1994, Haddawy et al. 2003, Langseth and Nielsen 2003, Yu 2007). Some recent work has allowed richer expressions of domain knowledge to be included into learning algorithms. Examples of this rich domain knowledge include qualitative monotonicities, which allow statements such as "higher values of *X* are more likely to produce higher values of *Y*" (Altendorf et al. 2005), and polyhedral knowledge sets (Fung et al. 2002), which describe regions of the input space that are known to contain certain classes of data points. Maclin and Shavlik (Maclin and Shavlik 1996) allow the user to interactively provide advice to a reinforcement learner using an imperative programming language. This rich domain knowledge is often incorporated using a constraint-based approach. Since most learning algorithms involve an optimization problem either during training or during inference, this domain knowledge is added to the model by incorporating these constraints during the optimization (Altendorf et al. 2005; Fung et al. 2002).

However, the incorporation of rich feedback has several key characteristics that distinguish it from much of the existing work on incorporating domain knowledge. First, domain knowledge is typically elicited from a domain expert and then encoded in the machine learning algorithm by the algorithm designer; this step occurs *before* the machine learning system is deployed. In contrast, our paper investigates rich feedback provided by an end user *after* the machine learning system is deployed. This is for situations in which the system cannot begin learning until after it is deployed as it needs to customize itself to the end user, who will have a different set of preferences than the algorithm designer. Second, in our problem setting, the end user, unlike the algorithm designer, is not an expert in machine learning. Consequently, the end user needs to be able to express his or her corrective rich feedback in a form that can be integrated into the learning algorithm without requiring the end user to have a deep understanding of the details of the learning algorithm. Finally, rich feedback is provided in an interactive setting, which requires learning algorithms that can be re-trained quickly with the newly acquired knowledge and also requires learning algorithms to be responsive to the corrective feedback in order to encourage user interaction.

In this interactive, rich feedback setting, constraint-based approaches are also frequently used. CueTIP (Shilman et al. 2006) uses user corrections as constraints for handwriting recognition. In Culotta et al. (Culotta et al. 2006), constraints are used to aid in information extraction. Huang and Mitchell (Huang and Mitchell 2006) use user feedback as constraints to refine a clustering algorithm. Other approaches include learning a similarity metric from the rich user feedback (Fogarty et al. 2008), clustering documents using a set of user-selected representative words for each class (Liu et al. 2004), allowing the user to directly build a decision tree for the data set with the help of visualization techniques (Ware et al. 2001). In Experiment #2 and Experiment #3, we propose two additional approaches for incorporating rich user feedback into machine learning that are viable in this interactive setting.

2.3 Email Classification by Learning Systems

Our research takes place in the domain of email classification by a learning system. Various learning algorithms, trained by end users, have been developed to predict the set of categories or folders defined by users associated with email messages (Brutlag and Meek 2000; Cohen 1996; Shen et al. 2006; Segal and Kephart 1999; Segal and Kephart 2000). Furthermore, there have been some efforts to automatically classify emails, unsupervised by a user (Kushmerick et al. 2006; Dredze et al. 2006). Email classification shares a number of issues and problems with machine learning in general. The reported accuracy of all these algorithms indicates that email classification is a very challenging problem. 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. With the given low accuracy, some research has been directed at helping users understand how these classifications have been made (Pazzani 2000; Crawford et al. 2002a, 2002b) but these approaches do not allow explanations from users back to the system to improve subsequent prediction. All these challenges further motivated our interest in studying rich user feedback and its potential in improving machine learning algorithms.

3. EXPERIMENT #1: EXPLAINING SYSTEM BEHAVIOR AND GETTING FEEDBACK FROM USERS

Experiment #1 was a formative study. Its purpose was to investigate RQ 1 and RQ 2, namely how machine learning systems should explain themselves to end users, and what kinds of improvement feedback end users might give to the machine learning systems.

3.1 Design, Participants and Procedure

To maximize external validity, it was important to base our experiment on real-world data. 122 messages from a user's email (farmer-d), which had sufficient content for human and machine classification, were drawn from the publicly available Enron dataset (Klimt and Yang 2004). The original Enron user had categorized these email messages into four email folders: Personal (36 messages), Resumé (27 messages), Bankrupt (23 messages) and Enron News (36 messages).

We employed a qualitative "think-aloud" design in order to extract the richest possible data from the participants. We observed and videotaped their activities and comments throughout the experiment, as well as collecting their work products.

As the first step of our procedure, learning algorithms classified each email message. Then, three explanations of each result were generated: a Rule-based, a Keyword-based, and a Similarity-based explanation. The application of the classification algorithms and the generation of explanations, described in the next section, were all done off-line prior to the experiment. We used the outcomes of these two steps to make the participants' interactions with our low-fidelity prototype possible.

To allow as thorough investigation of users' potential as possible, it was important to encourage participants to express feedback freely. Because low-fidelity prototypes do not implement full functionality and avoid the impression of a "finished" product, paper-based low-fidelity prototypes are often used to encourage participant feedback (Rettig 1994). Thus, we used printouts of emails instead of an on-line display. A second advantage to this set-up was that it allowed for flexibility and ease of feedback. Using pens, printouts, and a big table to support spatial arrangements (Figure 1), participants could move papers around to compare them, scratch things out, draw circles, or write on them in any way they chose (Figure 2).

The participants were 13 graduate and undergraduate students (7 females, 6 males). All had previous experience using computers but did not have computer science backgrounds. All were native English speakers. An overview of participant backgrounds is given in Appendix A. The experiment followed a within-subject design, in which each participant experienced all three explanation paradigms. We counterbalanced learning effects in our design by randomizing the order of explanation paradigms that each participant experienced.

The experiment was conducted one participant at a time with a facilitator interacting with the participant and an observer taking additional notes. First, the participant was familiarized with thinking aloud. Next, he or she looked through 40 sample pre-classified email messages to become familiar with the folders and to develop an intuition for how new email messages should be categorized; this sample was kept the same across participants. At this point, the main task began, divided into three 15-minute blocks (one per explanation paradigm) of processing the remainder of email messages from the dataset.

For the main task, we randomized assignments of emails to explanation paradigms to avoid exclusive association of an email with just one paradigm. For each message, the facilitator handed a new printout to the participant, who decided whether the predicted folder classification was correct, reclassified the message if needed, and gave feedback to improve the classification if needed. The participants were told that an evolving "virtual email assistant" had been implemented, and that we wanted their help "in order to get these predictions working as well as they possibly can."

After each paradigm's 15-minute block, participants provided subjective self-evaluations of mental effort, time pressure, overall effort, performance success, and frustration level, based on standard NASA TLX questions (Hart and Staveland 1988). They also took a comprehension test for that paradigm. Finally, at the end of the study, they compared and ranked all three explanation paradigms in terms of overall preference, ease of understanding, and ease of feedback.

3.2 Materials: Explanations of the Learning Algorithms

We generated three different explanations to explore aspects underlying their respective understandability (RQ 1) and their impact on user feedback (RQ 2). We generated the explanations under the following three principles:

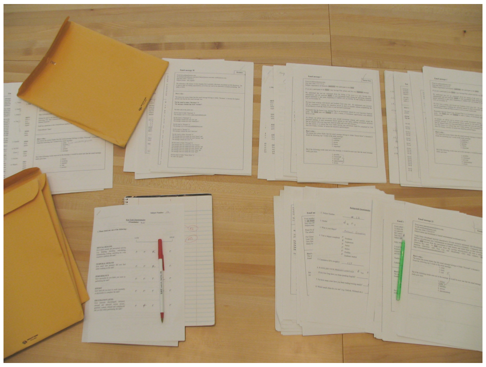
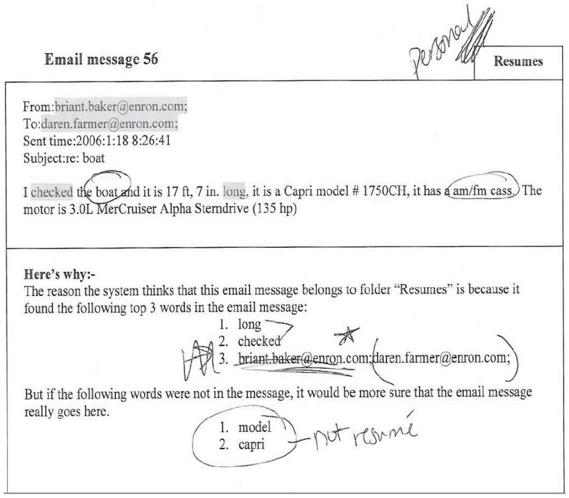


Figure 1: Lo-fi prototype set-up with pens, printouts, table.



Email message 56 personal | **Resumes**

From:briant.baker@enron.com;
To:daren.farmer@enron.com;
Sent time:2006:1:18 8:26:41
Subject:re: boat

I checked the boat and it is 17 ft, 7 in. long, it is a Capri model # 1750CH, it has a am/fm cass. The motor is 3.0L MerCruiser Alpha Sterndrive (135 hp)

Here's why:-
The reason the system thinks that this email message belongs to folder "Resumes" is because it found the following top 3 words in the email message:

1. long
2. checked
3. ~~briant.baker~~@enron.com;daren.farmer@enron.com;

But if the following words were not in the message, it would be more sure that the email message really goes here.

1. model
2. capri

not resumé

Figure 2: Example of participant feedback.

(Principle 1) Common algorithms: We focused on standard implementations of machine learning algorithms found in Weka (Witten and Frank 2005) that were viable for (a) generating explanations and (b) good performance in the email domain.

(Principle 2) Comprehensible but faithful explanations: It does not seem reasonable to provide end users with an explanation that requires a mathematical or statistical background. Instead, we sought to develop explanations that would be informal, yet accurate enough to engender useful mental models of this reasoning, analogous to "naïve physics" descriptions of qualitative physics.

(Principle 3) Concrete explanations: The explanations were required to be in terms of specific features that were visible in the current email message. In machine learning terms, each *feature* is an attribute that helps describe an email. In a bag-of-words approach, which is characteristic of our algorithms, the words that appear in emails are considered to be individual features.

3.2.1 Learning Algorithms and Training

We chose two learning algorithms to satisfy Principle 1: the Ripper rule-learning algorithm (Cohen 1996) and the Naïve Bayes algorithm (Mitchell 1997). These algorithms have been widely applied for email classification (e.g., Cohen 1996; Dalvi et al. 2004; Shen et al. 2006). Additionally, these algorithms can easily produce explanations of their results, which satisfies Principles 2 and 3. To obtain a prediction for each of the 122 email messages, we performed a stratified 5-fold cross-validation.

Prior to training, each email message was preprocessed to remove headers and common "stop" words. The remaining words were stemmed by Porter's method to remove word endings (Porter 1980). Each email message was then represented as a Boolean vector with a Boolean feature for each observed email sender (the From field), one Boolean feature for each observed set of email recipients (the union of the From, To, CC, and BCC fields, in essence identifying the "team" of people to which the message relates (Shen et al. 2006)), and one Boolean feature for each distinct word observed in the Subject and Body fields.

Ripper learns a set of classification rules. The rules are ordered by class but unordered within class. Hence, to make a prediction, Ripper first applied the rules for the least frequent class (Bankrupt in our dataset). If one of these rules matched the email message, it was classified as Bankrupt. Otherwise, Ripper moved on to the rules for the next most frequent class (Resume), and so on. There were no rules for the most frequent class (Enron News); it was treated as the default if none of the rules for the other classes matched.

Naïve Bayes estimates the probability of each folder given the email message. Let the random variable $F \in \{f_1, f_2, ...f_m\}$ represent a folder that the email message can be filed under. The variable F can take on m possible values, corresponding to the m folders. Additionally, let the vector $W = (w_1, w_2, ...w_n)$ be an email message where each component w_i represents a feature of the email. More precisely, each w_i is a Boolean variable indicating the presence or absence of a particular word in the email. The goal of the Naïve Bayes algorithm is to calculate the posterior probability $P(F = f_k \mid W)$. The Naïve Bayes algorithm simplifies the calculation of this posterior probability by making the assumption that probability of the each feature w_i is conditionally independent given the folder F. The mathematical details of the Naïve Bayes algorithm are shown in Figure B1 in Appendix B.

The overall accuracy of the predictions was not particularly high: 60% for Naïve Bayes and 75% for Ripper when used to classify the entire set of emails. We would have preferred higher accuracy and equal accuracy between algorithms. Still, high accuracy was not required to answer our experiment's research questions, and our analysis takes accuracy differences into account.

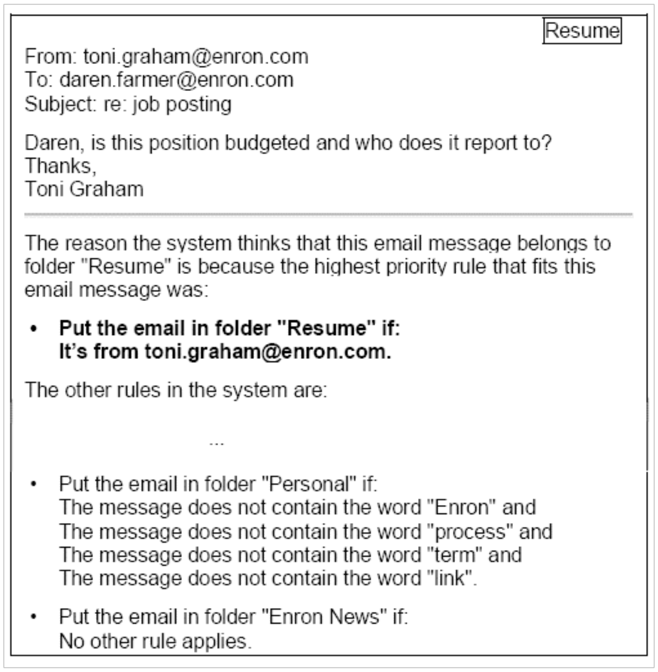
3.2.1 Generating Explanations

We generated the Rule-based explanations (Figure 3) by showing the rule, learned by the Ripper algorithm, that made the classification. That rule was listed above all other possible rules in the explanation.

We generated the Keyword-based and Similarity-based explanations from the learned Naïve Bayes classifier algorithm, as follows.

Consistent with our third design principle ("visible words only"), the Keyword-based explanations (Figure 4) listed and highlighted up to five words *present in the email message* having the largest positive weights as well as up to five words *present in the email message* having the most negative weights. (As we will discuss later, users found this latter set of "negative" words counter-intuitive. They are the words whose presence in the message *reduces* the certainty of the classifier in the sense that the classifier would be more confident if these words did *not* appear.)

The Similarity-based explanations (Figure 5) showed the training email message that, if deleted from the training set, would have most decreased the score. This was typically the training example most similar to the email message being classified. This example was displayed and up to five words with the highest weights that appeared in both the example and the target email message were outlined to draw them to the attention of the user.



Resume

From: toni.graham@enron.com
To: daren.farmer@enron.com
Subject: re: job posting

Daren, is this position budgeted and who does it report to?
Thanks,
Toni Graham

The reason the system thinks that this email message belongs to folder "Resume" is because the highest priority rule that fits this email message was:

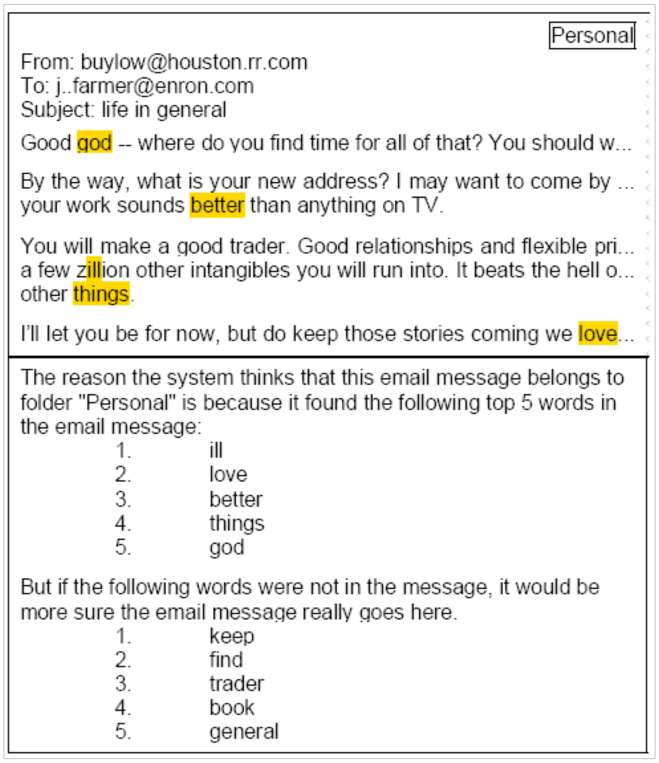
- **Put the email in folder "Resume" if:**
  It's from toni.graham@enron.com.

The other rules in the system are:

...

- Put the email in folder "Personal" if:
  The message does not contain the word "Enron" and
  The message does not contain the word "process" and
  The message does not contain the word "term" and
  The message does not contain the word "link".
- Put the email in folder "Enron News" if:
  No other rule applies.

Figure 3: (Top): Email.
(Bottom): Rule-based explanation excerpt.



Personal

From: buylow@houston.rr.com
To: j..farmer@enron.com
Subject: life in general

Good god -- where do you find time for all of that? You should w...

By the way, what is your new address? I may want to come by ...
your work sounds better than anything on TV.

You will make a good trader. Good relationships and flexible pri...
a few zillion other intangibles you will run into. It beats the hell o...
other things.

I'll let you be for now, but do keep those stories coming we love...

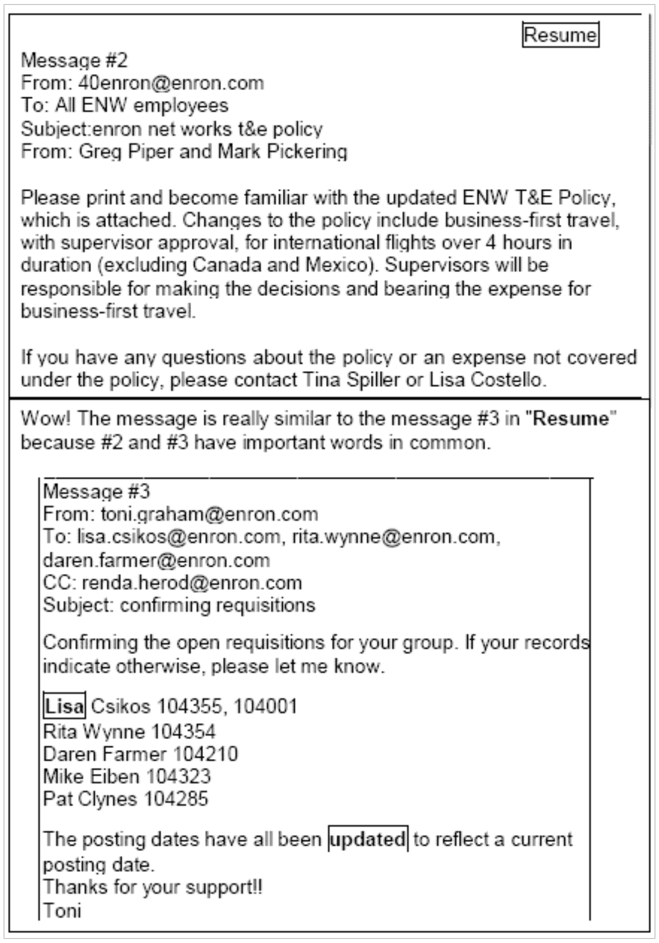
The reason the system thinks that this email message belongs to folder "Personal" is because it found the following top 5 words in the email message:

1. ill
2. love
3. better
4. things
5. god

But if the following words were not in the message, it would be more sure the email message really goes here.

1. keep
2. find
3. trader
4. book
5. general

Figure 4: (Top): Excerpt from email.
(Bottom): Keyword-based explanation, supplementing the highlights in the email.



Message #2
From: 40enron@enron.com
To: All ENW employees
Subject:enron net works t&e policy
From: Greg Piper and Mark Pickering

Please print and become familiar with the updated ENW T&E Policy, which is attached. Changes to the policy include business-first travel, with supervisor approval, for international flights over 4 hours in duration (excluding Canada and Mexico). Supervisors will be responsible for making the decisions and bearing the expense for business-first travel.

If you have any questions about the policy or an expense not covered under the policy, please contact Tina Spiller or Lisa Costello.

Wow! The message is really similar to the message #3 in "**Resume**" because #2 and #3 have important words in common.

Message #3
From: toni.graham@enron.com
To: lisa.csikos@enron.com, rita.wynne@enron.com, daren.farmer@enron.com
CC: renda.herod@enron.com
Subject: confirming requisitions

Confirming the open requisitions for your group. If your records indicate otherwise, please let me know.

Lisa Csikos 104355, 104001
Rita Wynne 104354
Daren Farmer 104210
Mike Eiben 104323
Pat Clynes 104285

The posting dates have all been **updated** to reflect a current posting date.
Thanks for your support!!
Toni

Figure 5: (Top): Excerpt from email.
(Bottom): Its Similarity-based explanation.

3.3 Analysis Methodology for Qualitative Data

We analyzed the think-aloud data and questionnaire comments using two coding schemes. In the first coding scheme, we coded all user utterances with the goal of determining the reaction of users to the explanations to answer research question RQ 1. In the second coding scheme, we performed a more detailed analysis of only the utterances that constituted negative comments about the explanations or suggested changes to the learning algorithms. The goal of the second coding scheme was to classify the users' feedback with respect to requirements for machine learning algorithms and the background knowledge required for the algorithms to be able to incorporate the feedback, in order to answer research question RQ 2. The first set of codes, along with a description and example are shown in the first three columns of Table 1. The second coding scheme is discussed in Section 3.4.4.

For both schemes, to ensure consistency in interpretation of the codes and when to use them, two researchers independently coded a small subset. They then iterated on this subset, further refining the codes and developing norms about how to apply them. After each iteration, agreement figures were calculated to test for coding reliability.

We calculated agreement in two ways. During the iterations, we used the Jaccard similarity index (Jaccard 1901) as a simple agreement measure. The Jaccard similarity index is calculated as the percentage of intersection of the codes divided by the union of all codes applied. For example, if one researcher gave the codes {Breakdown, Suggest Change} for one email and another researcher gave the codes as {Emotion, Suggest Change} for the same email, then the agreement was calculated as 1/3 (33%) as follows:

$$\frac{|\{Breakdown, SuggestChange\} \cap \{Emotion, SuggestChange\}|}{|\{Breakdown, SuggestChange\} \cup \{Emotion, SuggestChange\}|}$$

For the first coding scheme, the total agreement value was 81% for the first subset at the end of these iterations, which indicates high coding reliability. For the second coding scheme, the agreement was 82% after the iterations. At this point, the schemes were deemed robust enough, and the remaining data were then coded.

We also calculated the ending rater agreement using an extension of the Kappa statistic (Mezzich et al. 1981), which covers the non-exclusive application of codes. (Cohen's and Fleiss' Kappa in our situation are inappropriate as they assume mutually exclusive application of codes.) The results were consistent with the final Jaccard calculations: For the first coding scheme, Kappa was 0.80, for the second coding scheme Kappa was 0.86.

3.4 Results Of Experiment #1

Analyzing the video transcripts and questionnaires using the first coding scheme produced the counts shown in the final column of Table 1. (We will not discuss further the codes making up less than 1% of the total.)

Table 1: The first coding scheme.

Code	Description	Example from data	Count (% of total)
Breakdown	Expressing confusion or lack of understanding with the explanation of the algorithm.	I don’t understand why there is a second email.	41 (8%)
Understand	Explicitly showing evidence of understanding the explanation of the algorithm.	I see why it used "Houston" as negative	85 (17%)
Emotion	Expressing emotions.	It’s funny to me.	15 (3%)
Trust	Stating that he or she trusted the system.	I would probably trust it if I was doing email.	1 (<1%)
Expectation	Expressing an expectation for the system to behave in a certain way.	I hope that eventually the intelligent assistant would learn to give more reasons.	2 (<1%)
Suggest change	Correcting the explanations or otherwise suggesting changes to the system’s reasoning.	Different words could have been found in common, like "Agreement," "Ken Lay."	161 (32%)
Negative comment	Making negative comments about the explanation (without suggesting an improvement).	…arbitrary words: "energy" especially bad.	100 (20%)
Positive comment	Making positive comments about the explanation.	The Resume rules are good.	94 (19%)

3.4.1 Explaining to Users: Understandability

3.4.1.1 Which Paradigms Did They Understand?

According to the participants’ responses to the questionnaires, the Rule-based explanation paradigm was the most understandable (Table 2). This was corroborated by their verbal remarks: Rule-based explanations generated three times as many remarks indicating understanding and less than a tenth the remarks indicating breakdowns as either Keyword-based or Similarity-based explanations.

Differentiating between shallow and deep understanding reveals further insights. "Shallow understanding" in this context means that participants were simply able to make the classification decision the same way as the explanation paradigms. To gather data needed to assess this aspect, the questionnaires included an email without a classification or explanation, and asked the participants to categorize it to a folder based on what the paradigm would predict. Nearly all participants categorized it correctly in the case of Rule-based explanations and Keyword-based explanations, but only 4 of the 13 participants categorized the email correctly in the Similarity-based case.

"Deep understanding" implies understanding the *reasoning behind* the classification decision of the explanation paradigms. The questionnaires included an email with a classification but without the explanation, and participants were asked *why* the paradigm would classify an email the way it did. For the Rule-based explanation paradigm, a majority of participants answered by giving a rule, and some even managed to reconstruct a close version of the actual rule that was applied. For Keyword-based explanations, nearly all participants answered with keywords, even managing to identify correctly some of the keywords used in the actual example. However, only three participants answered even close to correctly for the Similarity-based case.

The combined evidence, from the participants’ opinions and their inability to explain or reproduce the Similarity logic, is thus quite strong that the Similarity-based explanations had a serious understandability problem.

Table 2: Participants' rankings from the written questionnaires. (Rank 1 is "understood the most.")

Explanation	Rank 1	Rank 2	Rank 3
Rule-based	9	2	2
Keyword-based	3	6	4
Similarity-based	1	5	7

3.4.1.2 What Factors Affected Understanding?

We investigated the factors that contributed to understandability via the Understand, Breakdown, and Negative Comments codes from the video transcripts and written questionnaire comments. Three factors stood out in affecting understanding of the system's behavior: understanding of the general idea of the algorithm, the Keyword-based explanations' negative keyword list, and appropriateness of word choices.

Regarding understanding of the algorithm, some participants expressed understanding of the algorithm by describing the essential strategy, as in the following two quotes. This enabled them to predict system behavior.

> *P6 (on Rule-based):* "I understand why it would just default to Enron, since that's what the rule is."

> *P1 (on Similarity-based):* "I guess it went in here because it was similar to another email I had already put in that folder."

In the case of the Keyword-based paradigm, some problems in understanding were caused by the negative keyword list. Nobody had anything positive to say about the inclusion of negative keywords in the explanation:

> *P6 (on Keyword-based):* "So what does this mean (referring to 2nd set of words)?"

> *P8 (on Keyword-based):* "I guess I really don't understand what it's doing here. If those words weren't in the message?"

Finally, appropriateness of the word choices seemed to have an effect on understanding, especially if they were felt by participants to be common words or topically unrelated:

> *P1 (on Similarity-based):* "'Day', 'soon', and 'listed' are incredibly arbitrary keywords."

3.4.1.3 Discussion: Understanding

In addition to the clear evidence of understandability problems for Similarity-based explanations, we would like to point out three results of particular interest.

First, although Rule-based explanations were consistently understandable to more than half the participants and, at least for this group of participants, seemed to "win" over the other two paradigms, note that about one-third of the participants preferred one of the other explanation paradigms. This implies that machine learning systems may need to support *multiple* explanation paradigms in order to effectively reach all of their users.

Second, Keyword-based explanations seemed to be reasonably understandable except for the negative keyword list, which our results suggest was a problem. There are several potential remedies. One possibility is that the negative keyword list could be explained in some different way to give users a better understanding of how the algorithm works. For example, instead of drawing attention to words with negative weights that are present in the email, the explanation could make use of the strongest negative weights associated with words that are *absent* from emails, since their absence increases the confidence of the learning algorithm. Another possibility is that the negative keyword list should be omitted from the explanation altogether.

Third, the *topical* appropriateness of word choices seemed particularly critical to participants' ability to predict and understand system behavior. This knowledge is too complex to be learned from only 122 email messages, but it could be possible in larger document collections; we will return to this point in Section 3.4.4.2.

3.4.2 Explaining to Users: Preferred Paradigms and Why

Following the understanding trends, participants' rankings favored the Rule-based explanations over the other two (Table 3). Still, nearly 50% of the participants chose a paradigm other than Rule-based as their favorite, so the Rule-based paradigm did not receive a clear mandate.

We expected preference to closely follow understanding trends, but analysis of the Positive Comments, the positive Emotion codes, and the questionnaire responses provided additional useful insights into factors that seemed to affect participants' preferences in positive ways. These remarks fell into four categories, three of which (approval of reasoning soundness, clear communication of reasoning, and perceived accuracy) tie at least somewhat to understandability.

Participants' approval of soundness of reasoning was remarked upon often. Also, clear *communication* of reasoning, which is distinctly different from the mere presence of sound reasoning, mattered to a number of our participants. For example, Participant 1's comment below is fairly representative of several about the reasoning itself, whereas Participant 10's comment exemplifies several comments specifically about communication of the reasoning:

> *P1 (on Keyword-based):* "The reasons seem like perfectly good reasons…this is a good reason why it shouldn't be in Personal."

> *P10 (on Similarity based)*: "I like this one because it shows relationship between other messages in the same folder rather than just spitting a bunch of rules with no reason behind it."

High accuracy, as perceived by the participants, was remarked upon often. (We will return to the influence of *actual* accuracy in Section 3.4.3). For example:

> *P11 (on Rule-based):* "I think this is a really good filter. Put in Resume if it's from toni.graham"

> *P2 (on Similarity-based):* "Similarity was my favorite - seemed the most accurate, and took email addresses into account."

The fourth category was unexpected: Several participants appreciated Similarity-based explanations' less technical style of expression, a characteristic we inadvertently introduced in our wording that emphasizes similarity ("Wow!"). This introduction of informality in the form of slang produced a number of Positive Comments for that explanation paradigm, pointing out possible benefits from relaxing the language style used in explanations. For example:

> *P1 (on Similarity-based): "*I also appreciate how the computer is excited about its decision... It's funny to me ... Told you, in conversational form, why it was similar."

> *P10 (on Similarity-based):* "This is funny... (laughs) ... This seems more personable. Seems like a narration rather than just straight rules. It's almost like a conversation."

Table 3: Participants' rankings from the written questionnaires. (Rank 1 is "preferred the most.")

Explanation	Rank 1	Rank 2	Rank 3
Rule-based	7	4	2
Keyword-based	3	4	6
Similarity-based	3	5	5

3.4.3 Accuracy

As we have mentioned, the algorithms performed at different accuracy rates, with Ripper outperforming Naïve Bayes. (We define "accurate" as being in agreement with the original Enron user who owned the email.) This suggests that Ripper may be a better choice than Naïve Bayes for accuracy in this type of situation. As to our experiment, accuracy rate was not statistically predictive via linear regression of any of the ratings provided by participants, did not result in differences in participants' willingness to provide feedback, and did not affect their accuracy in doing so.

When the participants disagreed with the machine (28% of the time), participants were usually right (22%), but not always (6%). Also, both the machine and the participants disagreed with the original user 22% of the time, suggesting some knowledge possessed only by the original user and perhaps even some misfiling by the original user. Ultimately, the participant corrections brought the accuracy rates for all paradigms to almost identical levels: 71-72%, a surprising result suggesting that the performance of the algorithms *did not matter to accuracy* in the end.

As the preceding paragraph also points out, the participants were not perfect oracles. The error rate is consistent with earlier findings regarding end-user programmers' accuracy in serving as oracles when debugging, which have reported error rates of 5-20% (e.g., Phalgune et al. 2005). This range of error rates has been robust across studies, and suggests a similar level of "noise" that users' judgments would introduce into the learning algorithm's data. How to deal with this level of noise is an open research topic for machine learning.

3.4.4 The Users Explain to the System

What did participants think machine learning algorithms should change? For all three paradigms, we coded participants' feedback (Negative Comment and Suggest Change) along two dimensions. The rows of Table 4 identify the type of change and the columns identify the knowledge needed to handle the change.

Table 4: Types of participants' changes (in rows) that required various background knowledge (in columns).

	KB-English	KB-commonsense	KB-domain	KB-other	Total	%
1. Select different features (words)	70	64	25	16	175	53%
2. Adjust weight	11	11	4	13	39	12%
3. Parse or extract in a different way	7	17	10	0	34	10%
4. Employ feature combinations	9	5	2	1	17	5%
5. Relational features	0	9	5	0	14	4%
6. Other	3	12	4	33	52	16%
Total	100	118	50	63	331	
%	30%	36%	15%	19%		

3.4.4.1 Participants' Suggestions by Type

The rows of Table 4 categorize the participants' suggestions into six types. The types of feedback were not independent of the explanation paradigms: some paradigms seemed to encourage participants to think along the lines of particular types of feedback. We will point out these influences along the way whenever a paradigm represents at least 50% of a category.

The first type, selecting different features, was the most widespread type of feedback, for all three explanation paradigms. Words that appear in an email are considered to be individual features by the machine learning algorithms. Each feature has a weight that influences its contribution to determining a particular classification. More than half of all feedback referred to either adding a new feature for the algorithm to consider or removing a feature from consideration, such as:

> *P13 (on Rule-based)*: "It should put email in 'Enron News' if it has the keywords 'changes' and 'policy'. I put down some keywords that I noticed."

The second type was comprised of suggestions in which participants agreed that a particular feature was worthy of note, but wanted to change its weight or importance. Participants' reactions to Keyword-based explanations generated 69% of the feedback of this type, perhaps because of the feature-focused nature of the Keyword-based paradigm. Some participants' suggestions for changing feature weight or importance involved adjusting the weight on features in a general sense, such as P8's below. Other participants, such as P7 and P1, flipped a weight from negative to positive (or vice versa), or focused on the frequency of the word occurrence in the email, akin to term weighting in information retrieval. Finally, participants such as P4 made other adjustments to ordering of relative importance.

> *P8 (on Keyword-based)*: "The second set of words should be given more importance."

> *P7 (on Keyword-based)*: "Keyword 'include' is not good for the second set. It should be in the first set of words."

> *P1 (on Rule-based)*: "'Bankruptcy' is here over and over again, and that seems to be the obvious word to have in here."

> *P4 (on Keyword-based)*: "I think that 'payroll' should come before 'year'."

The third type of feedback concerned parsing the text or extracting features from the text in a different way. Some participants suggested a different form of text parsing, such as P1 below. In the simplest case, this could be achieved by an improved stemming procedure (Porter 1980). In some cases, however, the suggested extraction operates on a structure such as a URL, such as P6. In this case, either the system would already need to know about URLs or else the user would need to define them (perhaps by giving examples). Participants such as P13 also suggested using the structure of the email to extract features, such as the "From" and "Subject" field. Finally, some participants such as P6 suggested new kinds of informative cues.

> *P1 (on Similarity-based):* "Different forms of the same word must be looked at."

> *P6 (on Similarity-based):* "I think it would be good to recognize a URL."

> *P13 (on Rule-based):* "Yea, I mean it has 'job' in the subject line (for sorting into Resumé folder)"

> *P6 (on Keyword-based):* "I think that it should look for typos in the punctuation for indicators toward Personal."

Feature combinations were the fourth type of user feedback. Participants pointed out that two or more features taken together could improve the prediction, especially when they were working with Similarity-based explanations, which generated 63% of the suggestions of this type:

> *P6 (on Similarity-based):* "I think it would be better if it recognized a last and a first name together."

> *P12 (on Keyword-based):* "I would think like 'authorize signature' or 'w-2 form'."

The fifth type of participant feedback suggested incorporating the use of relational features. In these cases, the participants used relationships between messages (threading) or organizational roles (chairman of Enron) to define a new feature. For example:

> *P6 (on Rule-based):* "I think maybe it should use the response and automatically put it in the folder with the message that was responded to."

> *P8 (on Keyword-based):* "This message should be in 'EnronNews' since it is from the chairman of the company."

The remaining feedback by users did not fall into the types above and we coded these as "Other". Most of this kind of feedback concerned changes to the learning algorithm itself. These included suggestions such as adding logical NOT to the rule language, eliminating the default rule in Ripper, requiring an equal number of positive and negative keywords in Keyword-based explanations.

There were also cases in which the real problem lay with the way the explanation was constructed, rather than with the learning algorithm:

> *P13 (on Similarity-based):* "Having 'points' being the only keyword, I mean that kind of sucks."

Although the algorithm actually used all of the keywords in the messages, the explanation only highlighted the shared words with the top weights. This suggests that an automated method for assimilating user feedback would need a component that could diagnose whether the perceived problem is due to approximations in the explanation or design decisions in the learning algorithm.

3.4.4.2 Participants' Suggestions by Knowledge Source

Table 4's columns categorize the participants' suggestions into four knowledge sources: knowledge of English, commonsense knowledge, domain-dependent knowledge, and other.

Almost a third (30%) of the participations' suggestions relied on knowledge of English (KB-English). We coded feedback in this category if the necessary knowledge could be learned from analysis of large document collections or obtained from other online resources (e.g., Wordnet (Miller 1995) or named-entity recognizers (Zhou and Su 2002)). For example:

> *P8 (on Rule-based):* "Does the computer know the difference between 'resumé' and 'resume'? You might have email where you're talking about 'resume' but not in a job-hiring sense."

> *P5 (on Keyword-based):* "Last names would be better indicators."

> *P1 (on Similarity-based):* "'day', 'soon' and 'listed' are incredibly arbitrary keywords."

Some knowledge might need to be manually encoded, but it could then be reused across many different organizations and applications (KB-Commonsense). For example, participants indicated that there are "families" of words that are work- or business-related, and also suggested topic-related words:

> *P4 (on Rule-based):* "'Policy' would probably be a good word that would be used a lot during business talk."

> *P1 (on Keyword-based):* "'Qualifications' would seem like a really good Resume word, I wonder why that's not down here."

Some knowledge is domain-dependent (KB-Domain). Some participant suggestions relied on knowledge specific to Enron. For example, in the following example, the machine learning system would need to know that Ken Lay was the CEO of Enron and that as such he carries special importance to Enron employees, would need to know the implications of being an Enron employee, and so on. Such knowledge would need to be encoded separately for each organization.

> *P11 (on Similarity-based):* "Different words could have been found in common like 'Agreement', 'Ken Lay'."

All remaining feedback was coded KB-Other. This usually occurred when a participant's comment was not specific enough to suggest the underlying knowledge source.

3.5 Issues Raised By Experiment #1 for Investigation by Experiments #2 and #3

Currently most learning systems take only a simple form of feedback into account: whether the prediction was right or wrong, or what the correct prediction should have been. *How* rich user feedback could be incorporated into existing algorithms has received little attention previously. Further, the effects of incorporating rich user feedback on accuracy require investigation. For example, one issue is that our participants were not perfect, and sometimes their feedback was erroneous; introducing such errors into the system's reasoning might make the algorithm perform worse rather than better. To shed light upon these issues, which are expressed by RQ 3, we conducted two off-line follow-up experiments, Experiment #2 and

Experiment #3.

Specifically, consider how the user feedback reported in Experiment #1 might be incorporated directly into the algorithms, i.e., how machine learning algorithms might be changed permanently to do as the users suggested in those comments without complex *in situ* reasoning advice from users. Consider Ripper and Naïve Bayes, the widely used algorithms that we used in Experiment #1. For these two algorithms, Type codes 1, 2, and 4 (features, weights, and feature combinations) as supported by KB-English are conducive of direct assimilation. These types of user feedback accounted for 27% of the 331 suggestions. Similarly, supporting these three type codes by KB-Commonsense rather than solely by KB-English is also possible but is more challenging, due to the difficulty of obtaining the relevant common sense topic models. Type codes 1, 2, and 4 that relied on KB-Commonsense accounted for an additional 24% of the suggestions. In contrast, the remaining type codes appear to require substantial additional machine learning research in feature extraction, relational learning, and user interfaces for providing application-specific feature specifications. Thus, a little over half of the participants' suggestions appear potentially viable for incorporation into these two particular algorithms.

Now consider the second issue, whether incorporating this feedback would improve accuracy. To make consideration of this issue tractable, we narrowed our focus to two of the feedback types, namely weight and keywords changes, without regard to knowledge source. Thus, we assumed that either the knowledge sources would be available, or else that the users themselves would serve as the knowledge source and would enter the changes interactively during reasoning.

We implemented the changes necessary for these two feedback types in the algorithms of Experiments #2 and #3, thereby covering 65% of the user feedback given. We used Naïve Bayes instead of Ripper as the underlying machine learning algorithm for Experiments #2 and #3. This choice was due to two main reasons. First, in Section 3.4.1.3, we highlighted the need to support multiple explanation paradigms. Naïve Bayes is capable of supporting all three of our explanation paradigms. Ripper, on the other hand, is most suited for a rule-based paradigm. Second, 64% of the user feedback in Table 4 (from type codes 1 and 2) correspond to feedback about keywords. Incorporating keyword-based feedback is much more natural for Naïve Bayes than for Ripper as automatically modifying the scoring functions for feature selection and pruning in Ripper is substantially more complex than for Naïve Bayes. We followed two approaches for incorporating the rich feedback into machine learning algorithms, namely, a constraint-based approach (Experiment #2) and a co-training approach (Experiment #3).

Our new algorithms allowed empirical exploration of the last part of our research question RQ 3, namely whether incorporating rich user feedback would actually improve accuracy. We evaluated both approaches retroactively using as inputs the keyword-based and similarity-based feedback our Experiment #1 participants had provided using a low-fidelity prototyping approach. In Sections 4 and 5, we report our results of the constraint-based and user co-training approaches, respectively, with accuracy comparisons against two baseline algorithms. Participant-by-participant details of the evaluations are given in Appendix C.

4. EXPERIMENT #2: USING USER FEEDBACK AS CONSTRAINTS

One possible approach is to incorporate rich user feedback of type codes 1 and 2 into existing algorithms as user constraints (e.g., "there should be a positive weight on feature X"). From the constraints, the corresponding weights can be automatically modified in the Naïve Bayes classifier.

From the user's point of view, this approach is appealing for three reasons. First, with the constraint-based approach, the system can heed the user's feedback right away. It does not require a large set of corrections for the feedback to begin making a difference. The reason the system can be so responsive is that, instead of merely converting the user feedback into an additional training example, the constraint-based approach can employ the rich user feedback directly in the actual machine learning algorithm by translating it into constraints. Second, the constraint-based approach is easily understandable and explainable: because the user's feedback is translated into constraints, the constraints the system is trying to heed can be explained much as the user originally expressed them. Third, constraints offer flexibility since they can be varied according to their "hardness." Some constraints that the user specifies may be hard constraints that the system *must* satisfy. Some constraints can be softer and should be considered by the system if possible but are not absolutely necessary.

4.1 The Constraint-Based Algorithm

We implemented the following three types of constraints:

Constraint type 1 (hard): If the participant reduced the weight or removed the word proposed in the explanation, the proposed word was considered vague or unimportant. The word was removed from the feature set.

Constraint type 2 (soft): If the participant increased the weight of a word, regardless of whether it was proposed in the explanation or not, the proposed word was assumed to be a strong indicator as to the email's folder. We incorporated this form of feedback by adding a constraint that forces the weight on the word to be positive, which made the word more important for the user-assigned folder than for other folders. In our implementation, we also increased the effect of this constraint by requiring the weight of the word to hold above some amount $\delta \geq 0$. Figure B2 in Appendix B illustrates the mathematical details of this second constraint type.

Constraint type 3 (soft): For the third constraint type, we assumed that words that had their weights increased by the user were more significant indicators of the user-assigned folder than other words appearing in email messages in that folder. In order to describe this constraint at a high level, we need to quantify the *predictiveness* of a word for the user-assigned folder. This predictiveness is defined as the probability of the email being assigned to the user-specified folder given that the word is present in the email message. We require that there should be a gap between the predictiveness of a user-selected word and the predictiveness of the most informative words for that folder that were not selected by the user. Thus, for each user-selected word, we added a set of 10 constraints to force the user-selected words to exceed by some amount $\theta \geq 0$ the predictiveness of the top 10 most informative words not selected by the user. The formal notation for constraints of type 3 are shown in Figure B3 in Appendix B.

The δ and θ parameters above control the hardness of the constraint and can take on any value between 0 and 1. The lower the value of these parameters, the softer the constraint. We varied these values in order to investigate the influence of the constraint parameter values on the accuracy of the classification. However, in this experiment, we observed very little difference in classification accuracy for different δ and θ values. As a result, we report results using parameter values of (δ =0, θ=0), (δ=0.4, θ=0), (δ =0, θ=0.4), (δ =0.4, θ=0.4), and (δ=0.8, θ=0.8) which are representative of the parameter values we tried. We chose not to investigate values over 0.8 as there was a danger of making the constraints too aggressive. (With θ values over 0.8, the risk was too high of over-generalizing spurious constraints over all the data, resulting in a problem in machine learning known as overfitting.)

Constraints of type 1 were incorporated into Naïve Bayes by simply excluding the feature from the classifier. The remaining two constraints types were incorporated into the parameters of the Naïve Bayes classifier. The standard method for setting the parameters for Naïve Bayes is to calculate the parameters from the training data using a process known as maximum likelihood estimation. We were able to incorporate the constraints by converting the maximum likelihood estimation procedure into a constrained optimization problem as shown in Figure B4 in Appendix B. In our experiment, this optimization was performed by the non-linear programming solver in Lindo (Lindo 2007).

4.2 Design, Procedure, and Data Sets

To evaluate the constraint-based approach, we compared it against a baseline approach representing the traditional way user feedback is incorporated into machine learning: a simple Naïve Bayes algorithm that takes into account only the corrected folder assignments provided by users in a training set ("this should be in Personal"), but not the suggestions' rich content ("because it's from Mom"). We refer to this simple Naïve Bayes algorithm as the "simple online training algorithm".

For the email data, we divided the emails from the Enron farmer-d dataset into three sets: the *training set*, the *feedback set* and the *evaluation set*. The training set contained emails that were used to train the machine learning classifiers. Emails in the feedback set were the ones that had been presented to participants in the main experiment. The final set, known as the evaluation set, consisted of emails that had not been previously seen by either the machine learning classifiers or the participants. We restrict the training, feedback and evaluation datasets to only include emails from two folders (Personal and Enron News). This restriction was needed for two main reasons. First, the Bankrupt and Resume folders have a very small number of emails. As a result, the evaluation set would only have a handful of emails from these folders, or in the worst case, none at all. The accuracy results reported on the evaluation set would have been dominated by the accuracy of filing emails to the Enron News and Personal folders. Second, this two-folder restriction was necessary to allow a head-to-head comparison between the constraint-based approaches in this section with the co-training approaches in Section 5. The co-training approaches in Section 5 require a large amount of unlabeled data; hence we needed the evaluation set to have a large amount of unsorted emails from all folders. We addressed this problem in Section 5 through the two-folder restriction. With the two-folder restriction in place, the training set contained 50 messages, the feedback set contained 11 messages[2], and the evaluation set contained 371 messages (88 emails in Enron News, 283 in Personal).

For the feedback data itself, we harvested the rich feedback participants gave in Experiment #1, and applied it to our new algorithms. We used their keyword-based feedback and their similarity-based feedback as inputs into our algorithms. (We did not use their rule-based feedback, since we did not build a rule-based algorithm, as we have already explained.) Participants were constrained by time, and since some participants were faster than others, some were able to provide feedback on more emails than others did. (See Appendix C1 for the exact number of emails for which each participant gave feedback.)

4.3 Results of Experiment #2

Figure 6 summarizes how the various algorithms fared using the participants' keyword-based feedback. Figure 7 shows a similar graph of how the algorithms performed using the participants' similarity-based feedback. (Full details of the evaluation are given in Appendix C). As the graphs show, there was little difference in accuracy between the various constraint parameters applied. On average, the constraint-based approach performed worse than the simple online training approach on both the keyword-based and similarity-based participant feedback.

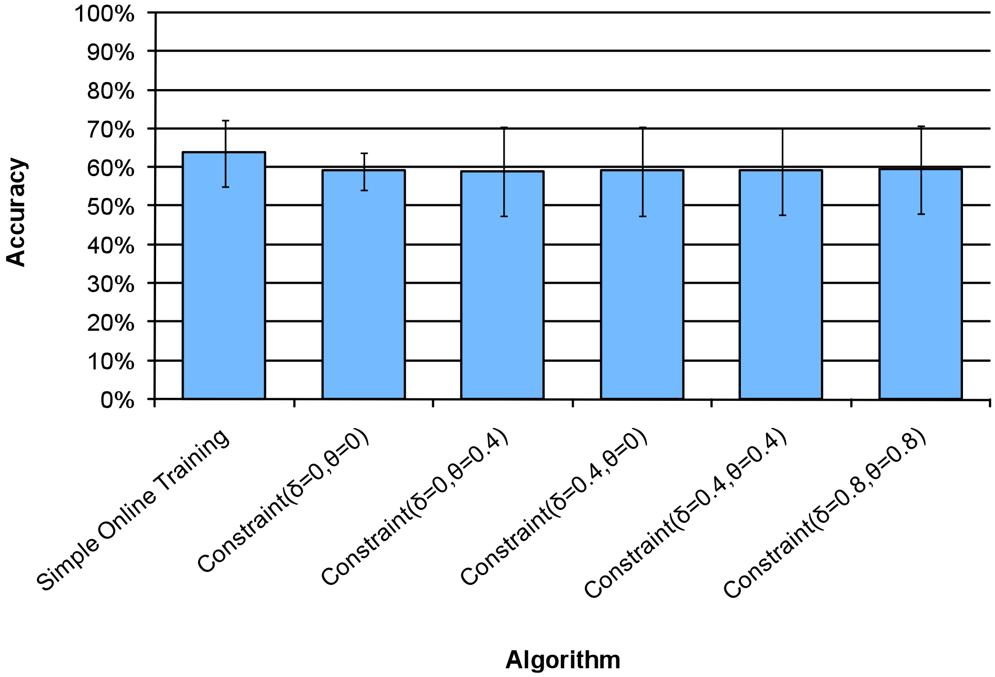


Figure 6: Percentage of emails the constraint-based algorithm (using different "softness" settings) sorted correctly after incorporating participants' rich feedback that they had given in Experiment #1 with keyword-based explanations. For comparison purposes, the simple online training algorithm (leftmost) uses participants' label feedback only, not their rich feedback. (Bars indicate standard error).

[2] Note that Appendix C1 lists the number of email messages out of the four Enron folders of Bankrupt, Resume, Enron News and Personal that received feedback from the participants. As mentioned in this paragraph, the algorithms were evaluated only on the Enron News and Personal folders. Therefore, only a subset of the feedback set could be used.

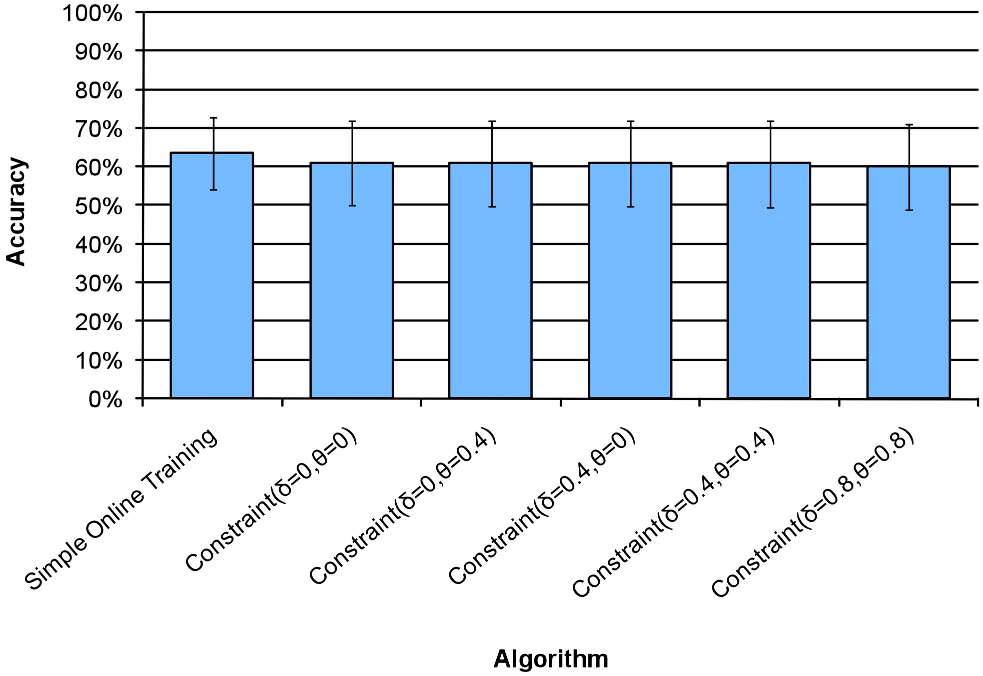


Figure 7: Percentage of emails the constraint-based algorithm ((using different "softness" settings) sorted correctly, after incorporating Experiment #1 participants' rich feedback that they had given in Experiment #1 with similarity-based explanations. For comparison purposes, the simple online training algorithm (leftmost) uses participants' label feedback only, not their rich feedback. (Bars indicate standard error).

4.4 Discussion of Experiment #2

The user's willingness to invest time providing information to an intelligent system can be viewed in terms of a cost-benefit-risk judgment by users, drawing on Attention Investment theory (Blackwell 2002). According to this theory, in weighing up whether to provide feedback, users' perceptions of costs, benefits, and risks each play key roles. One of the charges of a good user interface is to provide the user with the information needed to make assessments of costs, benefits, and risks that are reasonably close to actual costs, benefits, and risks. For example, some costs can be controlled through careful design of the intelligent interfaces to explain the system's reasoning and to accept feedback. Risk can arise due to the intelligent system's inability to make profitable use of the user's suggestions, because the feedback in some circumstances actually makes accuracy worse. Benefits can be gained by the system's ability to make effective use of the user's suggestions to improve its accuracy.

Results of Experiment #2 provide some evidence as to the amount of potential actual costs, risks and benefits to users in providing feedback to machine learning systems. Experiment #2's results were that the constraint-based approach had a lower accuracy on average than the simple online training approach by 5% in the keyword-based paradigm and approximately 3% in the similarity-based paradigm. As Appendix C4 details, the accuracy for the constraint-based variants was lower than that of the simple online training algorithm for all participants—except for participants 1, 6, and 11 for keyword-based user feedback and participants 5, 8, 11 and 13 for similarity-based feedback.

When we investigated why the performance decreased for the constraint-based algorithms, we discovered that the users' selection of which and how many features to feedback may increase the risk and reduce the benefit. This is due to several reasons. First, participants were inclined to propose the most significant words present in the folder (for example, "enron" for the Enron News folder) and this resulted in redundant constraints that had already been learned by the system. Giving feedback of this kind gave no additional benefits. Second, in some cases, the number of features proposed by participants made up only a small fraction of the total number of features considered by the system and therefore did not result in enough change to correct decisions made by the classifier. Finally, some of the constraints actually degraded the classifier by over-constraining the parameter space of Naïve Bayes, causing the learning algorithm to set its parameters to suboptimal values during training.

These results with the constraint-based approach show that its success depends on the hardness of the constraints. If the user feedback is low quality, we would like to soften the constraints but if the constraints are too weak, they will have almost no impact on the classifier. On the other hand, if the constraints are too hard then it could also lower the performance through overfitting. In practice, we found the hardness of the constraints to be very difficult to set. Overall, this approach resulted in reduced accuracy in Experiment #2. All these considerations may make the constraint-based approach unsuitable from an Attention Investment perspective (Blackwell 2002).

Of course, this is just one experiment with data from one group of users. Follow-up studies are needed to establish the generality of this algorithm's viability, or lack thereof, for incorporating user feedback in the form of constraints. In addition, the constraint-based approach only applies the constraints to a naïve Bayes classifier. For future work, we would like to incorporate user feedback in the form of constraints to other machine learning algorithms such as logistic regression, Ripper, and Support Vector Machines.

5. EXPERIMENT #3: USING USER FEEDBACK IN CO-TRAINING

As an alternative to the constraint-based approach, we explored a more complex machine learning approach that is similar in spirit to the co-training algorithm (Blum and Mitchell 1998), which is a commonly used algorithm in a sub-area of machine learning known as semi-supervised learning (Chapelle et al. 2006).

Semi-supervised learning is used when labeled data is scarce but unlabeled data is abundant. For example, consider the email sorting situation in which the user has assigned a small number of emails to folders while the majority of the emails in her inbox remain unsorted. The limited number of emails that have been assigned to folders is considered to be *labeled* data while the remaining unsorted emails are considered to be *unlabeled* data. Even though the majority of emails are unlabeled, they can still provide useful information for classification because they may form natural groupings based on the presence or absence of certain words. Semi-supervised learning aims to improve the performance of a learning algorithm trained on the small amount of labeled data by leveraging the structure of the unlabeled data.

The co-training (Blum and Mitchell 1998; Kiritchenko and Matwin 2001; Balcan et al. 2005) approach to semi-supervised learning employs two classifiers that work on the same data but have two different "views" of the data through independent sets of features. This approach operates under the assumption that the two classifiers produce the same classification even though they have different views. In our email example, the set of features for the first classifier consist of email addresses that appeared in the email header fields and in the body of an email message. The feature set for the second classifier includes words appearing in the subject line and the email body. These two feature sets are from the same emails but they are two independent sets. 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 an email message to a folder. The most confidently classified email message along with its folder assignment is then added to the training set for the next round of training. The standard co-training algorithm is described in pseudocode in Figure B5 in Appendix B.

5.1 User Co-training

We adapted the notion of co-training by regarding the *user* as if he or she were one of the classifiers in co-training, and a standard machine learning classifier such as Naïve Bayes or Ripper as the other classifier. We call this modified version of co-training *user co-training.* In order to treat the user as the second classifier, we developed a *user feedback classifier* that represents the user and treats the important words selected in the user feedback as a set of features for the specific folder to which the user assigns the email. Thus, associated with each folder *f* is a vector of important words $\boldsymbol{v_f}$ obtained by taking the union of all the important words selected by the user in the email messages placed into folder *f.* For the purposes of our experiment, these user-selected words were obtained during Experiment #1. The user feedback classifier uses this information to classify an email message by finding the folder with the highest number of words in $\boldsymbol{v_f}$ that appear in the message.

Note that unlike the machine learning classifier, the user feedback classifier is, by definition, trained only on the feedback set. Due to the limited number of user-selected words in Experiment #1, computing the classification confidence of the user feedback classifier resulted in too many folders having the exact same assignment probability. This situation would also arise in the real world, especially for during a user's early weeks of using a new email client. In order to avoid these tie-breaker conditions, we weighted the confidence of the user feedback classifier by the confidence of the machine learning classifier.

Pseudocode for the user co-training algorithm is shown in Figure B6 in Appendix B. The algorithm's variable $\boldsymbol{Confidence_f}$ was computed as the posterior probability of Naive Bayes. In this experiment, we required $\boldsymbol{Score}_{\mathrm{m}}$ to be positive. Due to the fact that the results did not change much for k > 10, we set the value of k to be 10.

In user co-training, the user has an equal influence in classification and training as the machine learning algorithm. We used Naïve Bayes as the standard machine learning classifier. We also included Ripper as a machine learning classifier in the co-training experiment, but the results did not show any substantial improvement. We therefore do not discuss it further here.

5.2 Design, Procedure, and Data Sets

We compared the user co-training approach against two baseline algorithms, Naïve Bayes (the simple online training algorithm from Experiment #2) and standard co-training.

As before, for the email data, we divided the emails from the Enron farmer-d dataset into three sets: the *training set*, the *feedback set* and the *evaluation set*. As mentioned earlier, we restricted these three data sets to contain emails from only the two folders of Enron News and Personal. This restriction was necessary as the co-training approaches require a large amount of unsorted emails from all folders in the evaluation set. Since the evaluation set did not have a substantial amount of emails belonging to the Bankrupt and Resume folders, we needed to restrict the training, feedback and evaluation datasets to only include emails from two folders (Personal and Enron News). The training set contained 50 messages, the feedback set contained 11 messages, and the evaluation set contained 371 messages (88 emails in Enron News, 283 in Personal).

Also as before, for the feedback data itself, we harvested the rich keyword-based and similarity-based feedback participants gave in Experiment #1, and applied it to our new algorithms. As in Experiment #2, the baseline algorithms made use of Experiment #1 participants' corrected folder assignments but not the participants' other suggestions.

To test the performance of the standard co-training algorithm in this experiment, we combined the emails in the training set and the feedback set into a data set and trained two classifiers employing separate feature sets. The set of features for the first classifier were email addresses that appeared in the From, To, Cc and Bcc fields and in the body of an email message. The feature set for the second classifier used all words appearing in the subject line and the email body. In our experiment, we set the confidence parameter θ to 0.95. (We experimented with θ values between 0.90 and 0.99 in increments of 0.01 and found little difference in the results.) We also set the number of iterations (i) equal to 10, because the results did not change significantly after 10 iterations.

5.3 Results of Experiment #3

Figures 8 and 9 compare user co-training against both the baseline algorithms of simple online training and standard co-training. As the figures show, the user co-training algorithm increased accuracy over the simple online training algorithm. Although a Wilcoxon test did not show significance using participants' similarity-based feedback, $z = -0.69$, $p = 0.2451$, the user co-training algorithm resulted in a highly significant improvement using participants' keyword-based feedback, $z = -2.53$, $p = 0.0057$.

It is clear that this improvement was not due to co-training in general—it was specifically tied to the *user* co-training algorithm. Standard co-training did not improve on accuracy over simple online training; in fact, it led to slightly decreased performance in both keyword-based and similarity-based. We hypothesize that the reason standard co-training did not do well is that the two independent feature sets do not, by themselves, produce a complete "view" of the data, thereby resulting in impoverished classifiers that can mislead each other in standard co-training.

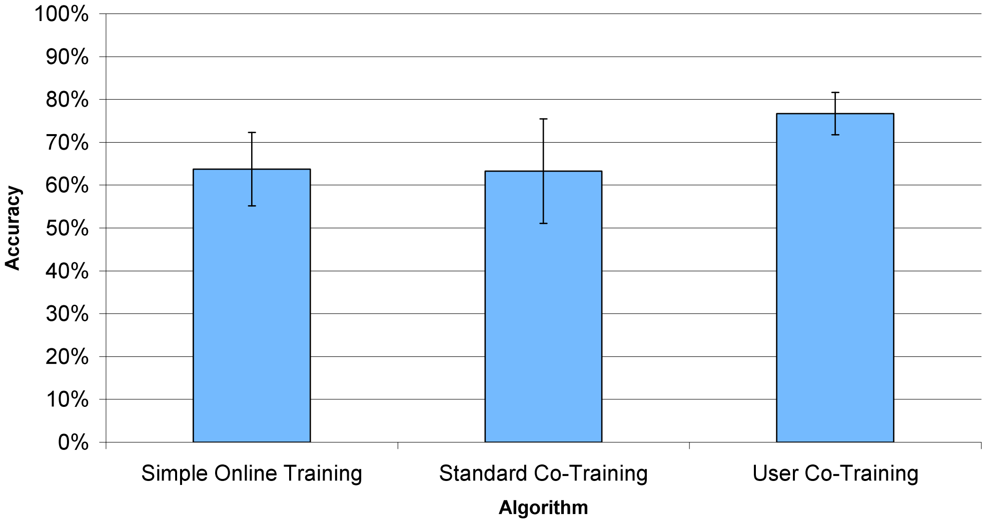


Figure 8: Percentage of emails the user co-training-based algorithm sorted correctly using keyword-based feedback

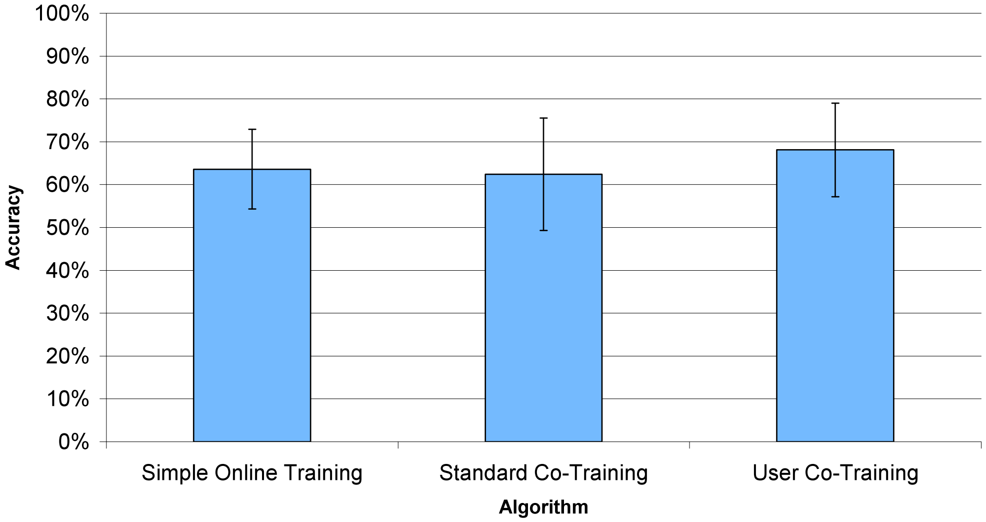
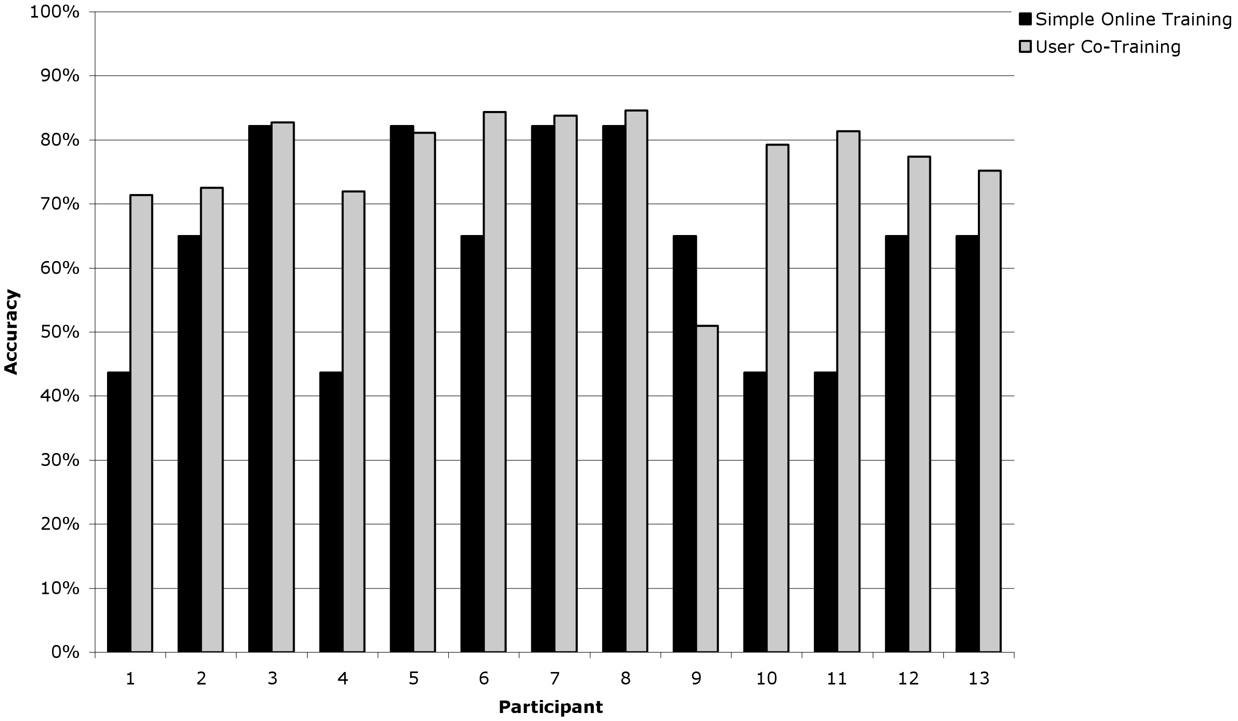


Figure 9: Percentage of emails the user co-training-based algorithm sorted correctly using the similarity-based feedback from Experiment #1, as compared to the baseline algorithms. (Bars indicate standard error).

To understand the particular circumstances when accuracy improvements could be expected, we investigated the user co-training approach in more detail for each participant. (Full results of the evaluation are given in Appendix C.) Figures 10 and 11 show the accuracy for the baseline online training approach compared with the user co-training approach for individual participants, taking into account keyword-based and similarity-based user feedback respectively. The user co-training approach had a higher accuracy than the simple online training approach in 11 out of 13 cases for the keyword-based paradigm and in 7 out of 12 cases for the similarity-based paradigm. In Figure 10, the results from participants 1, 4, 10 and 11 are particularly noteworthy. For these participants, the simple online algorithm's accuracy was low (below 50%) yet substantial improvements were made through applying the user co-training approach. Similarly, participants 2, 3 and 13 benefited for similarity-based feedback (in Figure 11). Thus, for both feedback input sets, user co-training's improvement was highest in cases when improvement was needed most, i.e., when the baseline approach performed badly because the label corrections alone did not yield enough feedback to correctly classify the emails. However, there were a few cases, i.e., participant 9 for keyword-based and participants 7, 8 and 11 for similarity-based, in which user co-training's performance was lower than the baseline approach by more than a marginal difference. Analysis of these participants' data did not reveal a pattern explaining these four participants' results.

5.4 Discussion of Experiment #3

The results of Experiment #3 indicate that the user co-training approach is a promising technique for incorporating user feedback and returning more benefit for the user's costs, especially if simple feedback through label corrections does not produce good accuracy. User co-training is also a fairly general approach; any machine learning algorithm can be combined with the user feedback classifier under the co-training framework. However, the user co-training approach makes an assumption that there is a fairly large amount of unsorted email in the user's inbox. A direction for future work is to develop new algorithms for incorporating rich user feedback in settings where this assumption is not met. In addition, the user co-training approach is not without risk. There are situations in which performance decreases by taking user feedback into account, possibly due to introduction of human errors into the learned information. An interesting extension to user co-training would involve modeling the reliability of the user and "softening" the rich user feedback when the machine learning system believes the user might be providing erroneous feedback. Further investigation is needed to evaluate how well user co-training works in real-world situations.



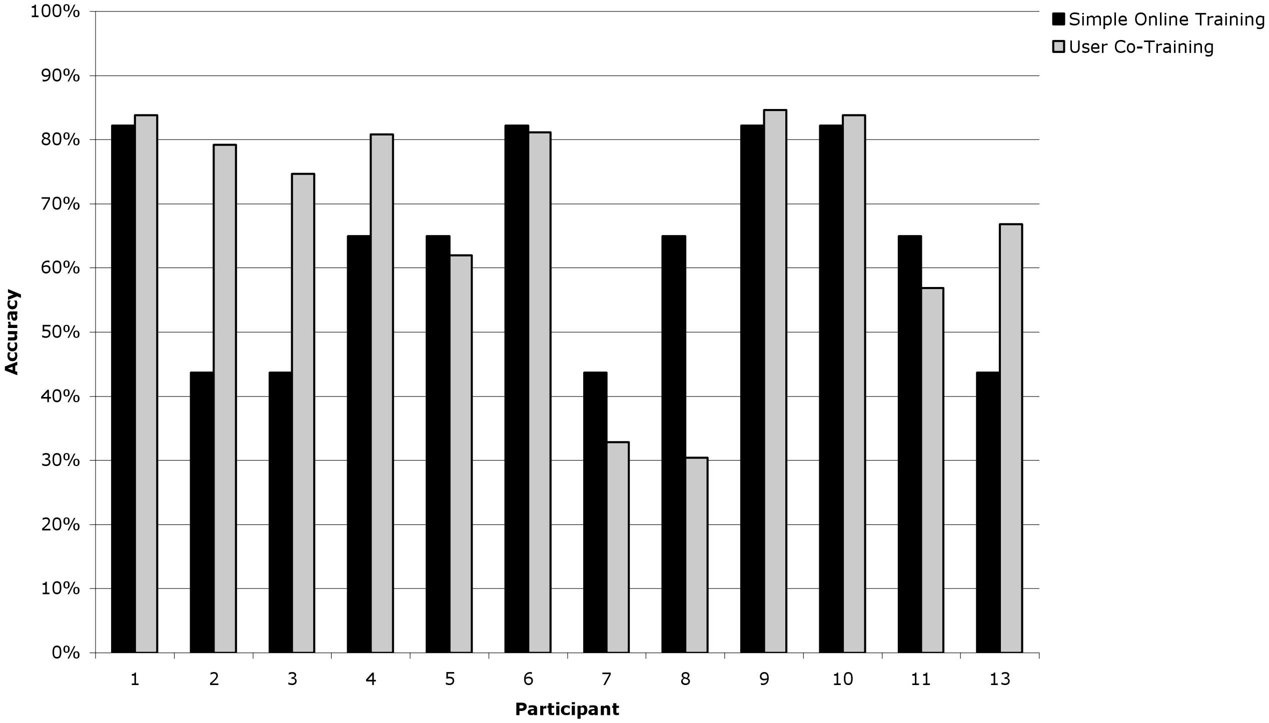


Figure 10 (keyword: top) and Figure 11 (similarity: bottom): Percentage of emails the user co-training algorithm sorted correctly for individual participants compared to the baseline approach.

While the results were that user co-training outperformed the constraint-based approach in Experiment #2, we caution against over-generalizing from these results. Experiment #3 showed that user co-training can be a viable approach for incorporating rich user feedback when its assumptions are met, but we do not claim that user co-training will always outperform a constraint-based approach. In fact, we believe that a constraint-based approach remains a feasible alternative to user co-training for incorporating rich user feedback since there are many possible learning algorithms to choose from and there is a tremendous amount of flexibility in how rich user feedback can be incorporated as constraints into these algorithms. We plan to investigate these different combinations of learning algorithms and constraints in future work.

6. CONCLUSION

The notion of humans interacting with a machine learning system, not just to override outcomes but rather to "rewire" the algorithms—changing the reasoning itself—is new. There are three components to this notion: (1) an intelligent system's ability to explain its reasoning to the user, (2) the user's reasoning corrections reflected in critiques and adjustments, and (3) the system's ability to make use of this rich user feedback in ways that are profitable for the system and ultimately for the user.

We have described three experiments that provide insights into all three of these components. Regarding the first component, an intelligent interface's ability to explain its reasoning to the user, we considered three explanation paradigms: Rule-based, Keyword-based, and Similarity-based. All were at least understandable enough that the participants were willing and reasonably effective at critiquing them. Rule-based explanations were the most understandable. Keyword-based were next, but the negative keywords list interfered. Similarity-based had serious understandability problems. But more important than which was the "winning" explanation paradigm is that there was no one winner: the lack of our participants' consensus suggests that multiple paradigms of explanations may need to be supported. We also reported data identifying the factors that won participant approval, which included perception of reasoning soundness, clear communication of reasoning, and informal wording ("Wow!").

Regarding the second component, the user's reasoning corrections made through critiques and adjustments, participants made a wide variety of reasoning suggestions, including reweighting features, feature combinations, relational features, and even wholesale changes to the algorithms. These suggestions were grounded in a variety of knowledge sources, such as knowledge of English, common-sense knowledge, and knowledge of the domain. Participants were more accurate than the machine—but they were not perfect, and occasionally made mistakes. This explicitly demonstrates the likelihood of rich user feedback introducing errors into the reasoning.

Regarding the third component, we were able to implement a subset of the participants' suggestions via two different algorithms, and empirically evaluated the resulting improvements in accuracy rates. The user co-training algorithm was the most effective, achieving substantial improvements in accuracy by taking into account the user suggestions. It had the highest accuracy on average, showing that although in some cases it may be outperformed by other approaches, in general it led to improvements in accuracy. In situations in which the simple online learning algorithm performed very badly, the user co-training approach led to the greatest improvement. These situations may be exactly the situations in which there is not enough training data present for standard machine learning approaches to succeed, and our results suggest that rich user feedback may provide the key to compensate for this lack of data.

Traditionally, machine learning approaches have been addressed primarily as an artificial intelligence problem, with users (when they are considered at all) as relatively passive spectators. We believe the design of intelligent systems needs to be viewed as a full-fledged HCI problem. Meaningful interaction involves both parties inseparably: approaches must be viable and productive for *both* the user and the system. This paper takes this view, considering both user and machine learning issues as inseparable parts of an intertwined whole. Overall, our results show evidence that intelligent interfaces can explain their reasoning and behavior to users, and that users in turn can provide rich, informative feedback to the intelligent system, and finally that machine learning algorithms can make use of this feedback to improve their performance. This suggests rich human-computer collaboration as a promising direction for machine learning systems to work more intelligently, hand-in-hand with the user.

ACKNOWLEDGMENTS

We thank the participants of our study. We also thank Kandace Bangert and Russell Drummond for their assistance. This work was supported by Intel, by NSF IIS-0133994, by NSF IIS-0803487, and by DARPA grant HR0011-04-1-0005 contract NBCHD030010.

REFERENCES

Altendorf, E., Restificar, E., and Dietterich, T. 2005. Learning from sparse data by exploiting monotonicity constraints. Proc. Uncertainty in Artificial Intelligence.

Balcan, M.F., Blum, A., and Yang, K. 2005. Co-training and expansion: Towards bridging theory and practice. Proc. NIPS.

Becker, B., Kohavi, R. and Sommerfield, D. 2001. Visualizing the simple Bayesian classifier. Information Visualization in Data Mining and Knowledge Discovery, 237-249.

Billsus, D., Hilbert, D. and Maynes-Aminzade, D. 2005. Improving proactive information systems. Proc. IUI, 159-166.

Blackwell, A. First Steps in Programming: A Rationale for Attention Investment Models. 2002. Proc. Human Centric Computing Languages and Environments.

Blythe, J. 2005. Task learning by instruction in Tailor. Proc. IUI, 191-198.

Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. Proc. COLT.

Bohanec, M. and Bratko, I. 1994. Trading accuracy and simplicity in decision trees. Machine Learning 15(3), 223-250.

Brutlag, J., Meek, C. 2000. Challenges of the email domain for text classification. Proc. ICML, 103-110.

Chapelle, O., Scholkopf, B., and Zien, A.. 2006. Semi-Supervised Learning. MIT Press, Cambridge, MA.

Chajewska, U. and Draper, D. 1998. Explaining predictions in Bayesian networks and influence diagrams. In Proceedings of the AAAI 1998 Spring Symposium Series: Interactive and Mixed-Initiative Decision-Theoretic Systems, 23-32.

Chklovski, T., Ratnakar, V. and Gill, Y. 2005. User interfaces with semi-formal representations: A study of designing argumentation structures. Proc. IUI, 130-136.

Clancey, W. J. 1983. The epistemology of a rule-based expert system – a framework for explanation. Artificial Intelligence, 20(3): 215-251.

Cohen, W. 1996. Learning rules that classify e-mail. Proc. AAAI Spring Symp. Information Access.

Cohn D. A., Ghahramani, Z., and Jordan, M. I. 1996. Active learning with statistical models. Journal of Artificial Intelligence Research 4, 129-145.

Craven, M, W. 1996. Extracting comprehensible models from trained neural networks. Ph.D. thesis. School of Computer Science, University of Wisconsin, Madison, WI.

Crawford, E., Kay, J., and McCreath, E. 2002a. An Intelligent Interface for Sorting Electronic Mail. Proc. IUI, 182-183.

Crawford, E., Kay, J., and McCreath, E. 2002b. IEMS – The Intelligent Email Sorter. Proc. ICML, 83-90.

Culotta, A. Kristjansson, T. McCallum, A. and Viola, P. 2006. Corrective Feedback and Persistent Learning for Information Extraction Artificial Intelligence, 170, 1101-1122.

Cunningham, S. J., Humphrey, M., and Witten, I. H. 1996. Understanding what machine learning produces. Technical Report 21, University of Waikato Department of Computer Science.

Cypher, A. (ed.) Watch What I Do: Programming by Demonstration, MIT Press, Cambridge, MA, 1993.

Dalvi, N., Domingos, P., Sanghai, M. S. and Verma, D. 2004. Adversarial classification. Proc. Intl. Conf. Knowledge Discovery and Data Mining, 99-108.

Dredze, M., Lau, T., Kushmerick, N.2006. Automatically classifying emails into activities, Proc. IUI.

Fails, J. A. and Olsen, D. R. 2003. Interactive machine learning. Proc. IUI, 39-45.

Fogarty, J., Tan, D., Kapoor, A., and Winder, S. 2008. CueFlik: interactive concept learning in image search. Proc CHI, 29-38.

Fung, G., Mangasarian, O. and Shavlik, J. 2002. Knowledge-based support vector machine classifiers. Proc. NIPS.

Glass, A., McGuinness, D. and Wolverton, M. 2008. Toward establishing trust in adaptive agents, Proc. IUI, 227-236.

Haddawy, P., Ha, V., Restificar, A., Geisler, B., Miyamoto, J. 2003. Preference elicitation via theory refinement. JMLR Special Issue on the Fusion of Domain Knowledge with Data for Decision Support, 4, 317-337.

Hart, S. and Staveland, L. 1988. Development of a NASA-TLX (Task load index): Results of empirical and theoretical research, in: Hancock, P. and Meshkati, N. (eds.), Human Mental Workload, 139-183.

Herlocker, J., Konstan, J. and Riedl, J. 2000. Explaining collaborative filtering recommendations. Proc. CSCW, 241-250.

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. Proc. CHI 1999, 159-166.

Huang, Y. and Mitchell, T. M. 2006. Text clustering with extended user feedback. Proc. SIGIR, 413-420.

Jaccard, P. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bulletin del la Société Vaudoise des Sciences Naturelles, 37, 547-579.

Johnson, W. L. 1994. Agents that learn to explain themselves. Proc. Twelfth National Conference on Artificial Intelligence, 1257-1263.

Kiritchenko, S. and Matwin, S. 2001. Email classification with co-training. Proc. Centre for Advanced Studies Conference, 8-17

Klimt, B. and Yang, Y. 2004. The Enron corpus: A new dataset for email classification research. Proc. European Conf. Machine Learning 2004, 217-226.

Kononenko, I. 1993. Inductive and bayesian learning in medical diagnosis. Applied Artificial Intelligence, 7, 317-337.

Kushmerick, N., Lau, T., Dredze, M., Khoussainov, R. 2006. Activity-centric email: a machine learning approach, Proc. AAAI.

Lacave, C., and Diez, F. 2002. A review of explanation methods for Bayesian networks. The Knowledge Engineering Review, 17, 2, Cambridge University Press, 107-127.

Langseth, H., and Nielsen, T. D. Fusion of domain knowledge with data for structural learning in object oriented domains. JMLR Special Issue on the Fusion of Domain Knowledge with Data for Decision Support, 4, 339-368.

Lieberman, H., (ed.) Your Wish is My Command: Programming By Example. 2001.

Lieberman, H. and Kumar, A. 2006. Providing expert advice by analogy for on-line help. Proc. Intl. Conf. Intelligent Agent Technology, 26-32.

Lindo Systems, Inc., 2007, http://www.lindo.com.

Liu, B. Li, X. Lee, W. and Yu, P. 2004. Text Classification by Labeling Words. Proc. AAAI.

MacKay, D. J. C. 1992. Information-based objective functions for active data selection. Neural Computation 4(4), 1992, 590-604.

Maclin, R. and Shavlik, J. 1996. Creating advice-taking reinforcement learners. Machine Learning, 22, 251-286.

McCarthy, K., Reilly, J., McGinty, L. and Smyth, B. 2005. Experiments in dynamic critiquing. Proc. IUI, 175-182.

McDaniel, R.G. and Myers, B.A. 1999. Getting more out of programming-by-demonstration. Proc. CHI, 442-449.

Mezzich, J. E., Kraemer, H. C., Worthington, D. R., and Coffman, G. A. 1981. Assessment of agreement among several raters formulating multiple diagnoses. Journal of Psychiatric Research, 16, 1, 29-39.

Miller, G. 1995. WordNet: A lexical database for English. Comm. ACM 38(11), 39-41.

Mitchell, T. 1997. Machine Learning. McGraw-Hill, Boston, MA.

Mozina, M., Demsar, J., Kattan, M., and Zupan, B. 2004. Nomograms for visualization of naive Bayesian classifier. Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases, 337-348.

Myers, B., Weitzman, D., Ko, A. Chau, D. 2006. Answering why and why not questions in user interfaces. Proc. CHI 2006, 22-27.

Oblinger, D., Castelli, V. and Bergman, L. 2006. Augmentation-based learning. Proc. IUI 2006, 202-209.

Pazzani, M. J. 2000. Representation of electronic mail filtering profiles: a user study. Proc. IUI 2000, 202-206.

Phalgune, A., Kissinger, C., Burnett, M., Cook, C., Beckwith, L. Ruthruff, J. 2005. Garbage in, garbage out? An empirical look at oracle mistakes by end-user programmers. Proc. Symp. Visual Languages and Human Centric Computing, 45-52.

Porter, M. 1980. An algorithm for suffix stripping. Program, 14(3), 130-137.

Poulin, B., Eisner, R., Szafron, D., Lu, P., Greiner, R., Wishart, D. S., Fyshe, A., Pearcy, B., MacDonell, C., and Anvik, J. 2006. Visual explanation of evidence in additive classifiers. Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAI).

Pu, P. and Chen, L. 2006. Trust building with explanation interfaces. Proc. IUI, 93-100.

Quinlan, J. R. 1993. C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann Publishers, Inc.

Rettig, M. 1994. Prototyping for tiny fingers. Comm. ACM 37(4), 21-27.

Ridgeway, G., Madigan, D., Richardson, T. and O'Kane, J. 1998. Interpretable boosted naive Bayes classification. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, 101-104.

Russell, S. J. 1991. Prior knowledge and autonomous learning. Robotics and Autonomous Systems, 8, 145-159.

Segal, R. and Kephart, J. 1999. Mailcat: an intelligent assistant for organizing e-mail. Proc. of the Third International Conference on Autonomous Agents.

Segal, R. and Kephart, J. 2000. Incremental learning in SwiftFile. Proc. International Conference on Machine Learning, 863-870.

Shen, J., Li, L., Dietterich, T. Herlocker, J. 2006. A hybrid learning system for recognizing user tasks from desk activities and email messages**.** Proc. IUI 2006, 86-92.

Shilman, M., Tan, D., and Simard, P. 2006. CueTIP: a mixed-initiative interface for correcting handwriting errors. Proc UIST, 323-332.

Stumpf S., Rajaram V., Li L., Burnett M., Dietterich T., Sullivan E., Drummond R., Herlocker J. 2007. Toward Harnessing User Feedback For Machine Learning. Proc IUI, 82-91.

Suermondt, H. J. 1992. Explanation in bayesian belief networks. Ph.D thesis, Stanford Univerity, Stanford, CA.

Swartout, W. R. 1983. Xplain: A system for creating and explaining expert consulting programs. Artificial Intelligence, 21, 285-325.

Tong, S. and Koller, D. 2002. Support vector machine active learning with applications to text classification. Journal of Machine Learning Research 2, 45-66.

Towell, G., and Shavlik, J. 1994. Knowledge-based artificial neural networks. Artificial Intelligence, 70, 119-165.

van de Merckt, T. and Decaestecker, C. 1995. Multiple-knowledge representations in concept learning. European Conference on Machine Learning (ECML), 200-217.

Ware, M., Frank, E., Holmes, G., Hall, M., Witten, I. H. 2001. Interactive machine learning: letting users build classifiers. IJCHS, 55, 281-292.

Wick, M. R. and Thompson, W. B. 1992. Reconstructive expert system explanation. Artificial Intelligence, 54(1-2):33-70.

Witten, I., Frank, E. 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Ed., Morgan Kaufmann, 2005.

Yu, T. 2007. Incorporating prior domain knowledge into inductive machine learning: its implementation in contemporary capital markets. Ph.D. thesis. Faculty of Information Technology, University of Technology, Sydney.

Zhou, G., Su, J. 2002. Named entity recognition using an HMM-based chunk tagger. Proc. Assoc. Comp. Linguistics, 473-480.

Appendix A: Detailed Participant Background

Participant Number	Gender	Major	Year	CGPA	Years of Email Experience
1	Female	English	Senior	3.85	10
2	Male	Business Admin	Junior	3.53	5
3	Male	Bio Eng	Senior	3.7	8
4	Female	Marketing	Junior	3.5	3
5	Male	Business	Senior	3.45	10
6	Male	Business Admin	Sophomore	3.82	7
7	Male	Food Science	N/a	2.5	8
8	Female	Bio Chem	Senior	3.91	4
9	Female	Animal Science	Sophomore	3.0	6
10	Female	Business Admin	Senior	3.15	9
11	Female	Business Admin	Junior	3.79	10
12	Female	Sport Sci	Freshman	2.3	10
13	Male	Mech Eng	Sophomore	3.28	8

Appendix B: Algorithm Details

B1. NAÏVE BAYES CLASSIFIER

The goal of the Naïve Bayes classifier is to compute $P(F = f_k \mid W)$

Using Bayes rule, we get:

$$P(F = f_k \mid W) = \frac{P(W \mid F = f_k)P(F = f_k)}{P(W)}$$

Assuming that the probability of each word w_i is conditionally independent given the folder *F*, the formula above can be rewritten as:

$$P(F = f_k \mid W) = \frac{P(W \mid F = f_k)P(F = f_k)}{P(W)} = \frac{\prod_{j=1}^{n} P(w_j \mid F = f_k)P(F = f_k)}{P(W)}$$

The optimal decision is the f_k that maximize the posterior probability $P(F = f_k \mid W)$. This is equivalent to finding out the maximum value of the ratio:

$$Ratio = \frac{P(F = f_k \mid W)}{P(F \neq f_k \mid W)} = \frac{P(W \mid F = f_k)P(F = f_k)}{P(W \mid F \neq f_k)P(F \neq f_k)} = \frac{\prod_{j=1}^{n} P(w_j \mid F = f_k)P(F = f_k)}{\prod_{j=1}^{n} P(w_j \mid F \neq f_k)P(F \neq f_k)}$$

The log of the ratio is:

$$\log\frac{P(F = f_k \mid W)}{P(F \neq f_k \mid W)} = \log\frac{P(w_1 \mid F = f_k)}{P(w_1 \mid F \neq f_k)} + \log\frac{P(w_2 \mid F = f_k)}{P(w_2 \mid F \neq f_k)} + \ldots + \log\frac{P(w_n \mid F = f_k)}{P(w_n \mid F \neq f_k)} + \log\frac{P(F = f_k)}{P(F \neq f_k)}$$

In the equation above, the Naïve Bayes classifier is transformed to a linear classifier with weight λ_{jk} for each word *j* and each email folder *k* ie.

$$\lambda_{jk} = \log\frac{P(w_j \mid F = f_k)}{P(w_j \mid F \neq f_k)}$$

The input to the Naïve Bayes classifier is an email message, which is converted to a vector of Boolean variables $W = (w_1,\ldots,w_n)$ where each component w_j was an indicator variable representing the presence of the *j*th word of the vocabulary in the email message. To predict the email folder under which the vector *W* would be filed, the classifier computed the folder *k* with the highest score ie.

$$\text{Predicted Folder} = \underset{\text{k}}{\text{argmax}}\ score(W,k) = \arg\max_{k} \sum_j \lambda_{jk} \cdot w_j$$

Figure B1: The mathematical details of the Naïve Bayes classifier

B2. CONSTRAINT REASONING DETAILS

Constraints of type 2 were represented by constraining weights to be positive as follows:

$$\lambda_{ik} > 0$$

$$= \log \frac{P(w_j = 1 \mid F = f_k)}{P(w_j = 1 \mid F \neq f_k)} > 0$$

$$= \log P(w_j = 1 \mid F = f_k) > \log P(w_j = 1 \mid F \neq f_k)$$

Where:

λ_{jk} is the weight associated with the word *j* and the index *k* of the user-assigned folder.

F is the random variable representing the folder

f_k is the value of the random variable *F* ie. the folder that the user assigned the email to

w_j is a Boolean variable indicating the presence or absence of the *j*th word.

If the log of the ratio was greater than 0, then the conditional probability in the numerator was greater than the conditional probability in the denominator. Therefore, if this inequality held, the probability of the word w_j being present in the email given that the user assigned the email to folder f_k was greater than the probability of the word w_j being present in the email given that the user assigned the email to any other folder than f_k. Consequently, the presence of the word w_j had more of an effect when classifying the email to folder f_k. To increase the effect of this constraint, we required the inequality to hold above some amount $\delta \geq 0$, i.e., $\log P(w_j = 1 \mid F = f_k) > \log P(w_j = 1 \mid F \neq f_k) + \delta$.

Figure B2: Constraint Type 2.

Constraints of type 3 were expressed as an inequality of the following form:

$$P(F = f_k \mid w_j = 1) > P(F = f_k \mid w_k = 1) + \theta$$

Where:

F is the random variable for the folder

f_k is the folder that the email was assigned to

w_j is a word selected by the user

w_k is one of the top 10 words appearing in the email message that was a strong predictor of the user-assigned folder for the email message but was not selected by the user.

θ is a parameter controlling the hardness of the constraint.

For each user-selected word, we added 10 constraints of this form. These constraints were determined by sorting the conditional probabilities $P(F = f_k \mid w_k = 1)$ for all possible words w_k and creating a constraint $P(F = f_k \mid w_j = 1) > P(F = f_k \mid w_k = 1) + \theta$ for each word w_k in the top 10 largest conditional probabilities in this sorted list.

Figure B3: Constraint Type 3

```
Maximize:
    The likelihood of the Naïve Bayes model on the all the samples in the training set and feedback set
Subject To:
    For each message in the feedback set,
        For all words $w_j$ that the user proposed or increased its weight in the message,
            $P(w_j = 1 | F = f_k) > P(w_j = 1 | F \neq f_k) + \delta$        [Constraint type 2]
            $P(F = f_k | w_j) > P(F = f_k | w_k) + \theta$                   [Constraint type 3]
Where:
    $f_k$ is the user assigned folder
    $w_k$ is one of the words that user did not select but is in the top 10 of $P(F = f_k | w_k)$
```

Figure B4: The Constraint-based algorithm.

B3. CO-TRAINING ALGORITHMS

```
Create two classifiers $C_1$ and $C_2$ based on the two independent feature sets.
Repeat $i$ times
    For each message in the evaluation set
    Classify the message with the two classifiers
    If classification confidence of any classifier is > $\theta$
        Add the classified message to the training data
    Rebuild $C_1$ and $C_2$ with the new training data
```

Figure B5: The standard co-training algorithm. In the Blum and Mitchell 1998 version of co-training, after training the two classifiers on the labeled data, the top *N* most confidently classified negative data points and the top *P* most confidently classified positive data points are added to the training set. A minor difference in our implementation is the fact that we add all data points that are classified above some confidence threshold *θ* to the training data.

```
For each folder $f$, create a vector $v_f$ to store the important words proposed by the user
For each message $m$ in the unlabeled set
    For each folder $f$,
        Compute $Confidence_f$ from the machine learning classifier
        $FolderScore_f$=# of words in $v_f$ appearing in the message $\times$ $Confidence_f$
    Find the folder $f_{max}$ that has the largest $FolderScore_f$ over all folders.
    Let $FolderScore_{other}$ be the max $FolderScore$ of the remaining folders (ie. excluding $f_{max}$)
    Save the message $Score_m = FolderScore_{fmax} – FolderScore_{other}$
Sort $Score_m$ for all messages in decreasing order
Select the top $k$ messages to add to the training set along with their folder label $f_{max}$
```

Figure B6: Our user co-training algorithm.

Appendix C: Detailed Results Comparing Learning Approaches to Take User Feedback into Account

C1. NUMBER OF MESSAGES WITH FEEDBACK

The number of emails processed by each participant:

	Participant												
Type of Feedback	1	2	3	4	5	6	7	8	9	10	11	12	13
Keyword-based	10	5	5	5	3	2	9	7	5	3	7	5	6
Similarity-based	9	5	8	7	5	4	7	6	4	2	5	0	4
Rule-based	8	1	3	4	3	1	2	5	1	1	4	2	9

C2. RESULTS FOR RULE-BASED USER FEEDBACK

The percentage of emails sorted correctly:

	Participant								
Algorithm	1	4	5	7	8	9	10	11	13
Simple Online Training	0.73	0.73	0.70	0.73	0.70	0.70	0.73	0.73	0.73
Standard Co-Training	0.72	0.70	0.70	0.72	0.70	0.70	0.72	0.70	0.70
User Co-Training	0.73	0.73	0.70	0.72	0.73	0.70	0.73	0.73	0.73

C3. RESULTS FOR KEYWORD-BASED USER FEEDBACK

The percentage of emails sorted correctly:

	Participant												
Algorithm	1	2	3	4	5	6	7	8	9	10	11	12	13
Simple Online Training	0.44	0.65	0.82	0.44	0.82	0.65	0.82	0.82	0.65	0.44	0.44	0.65	0.65
Standard Co-Training	0.32	0.73	0.83	0.32	0.83	0.73	0.83	0.83	0.73	0.32	0.32	0.73	0.73
User Co-Training	0.71	0.73	0.83	0.72	0.81	0.84	0.84	0.85	0.51	0.79	0.81	0.77	0.75
Constraint(δ=0,θ=0)	0.79	0.30	0.74	0.29	0.78	0.81	0.60	0.77	0.29	0.36	0.75	0.67	0.52
Constraint(δ=0, θ=0.4)	0.79	0.30	0.74	0.29	0.78	0.81	0.60	0.77	0.29	0.36	0.75	0.67	0.52
Constraint(δ=0.4, θ=0)	0.77	0.31	0.75	0.30	0.78	0.80	0.62	0.77	0.29	0.37	0.75	0.67	0.51
Constraint(δ=0.4, θ=0.4)	0.77	0.31	0.75	0.30	0.78	0.80	0.62	0.77	0.29	0.37	0.75	0.67	0.51
Constraint(δ=0.8, θ=0.8)	0.74	0.34	0.77	0.30	0.78	0.80	0.67	0.76	0.29	0.39	0.74	0.67	0.47

C4. RESULTS FOR SIMILARITY-BASED USER FEEDBACK

The percentage of emails sorted correctly:

	Participant											
Algorithm	1	2	3	4	5	6	7	8	9	10	11	13
Simple Online Training	0.82	0.44	0.44	0.65	0.65	0.82	0.44	0.65	0.82	0.82	0.65	0.44
Standard Co-Training	0.83	0.32	0.32	0.73	0.73	0.83	0.32	0.73	0.83	0.83	0.73	0.32
User Co-Training	0.84	0.79	0.75	0.81	0.62	0.81	0.33	0.30	0.85	0.84	0.57	0.67
Constraint(δ=0,θ=0)	0.74	0.31	0.39	0.40	0.67	0.81	0.31	0.74	0.80	0.74	0.76	0.64
Constraint(δ=0, θ=0.4)	0.74	0.31	0.39	0.40	0.66	0.81	0.31	0.74	0.80	0.74	0.76	0.64
Constraint(δ=0.4, θ=0)	0.75	0.31	0.39	0.40	0.64	0.81	0.31	0.75	0.80	0.75	0.75	0.64
Constraint(δ=0.4, θ=0.4)	0.75	0.31	0.39	0.40	0.64	0.81	0.31	0.75	0.80	0.74	0.75	0.64
Constraint(δ=0.8, θ=0.8)	0.75	0.31	0.41	0.41	0.52	0.81	0.32	0.76	0.80	0.75	0.75	0.63