Improving Automated Email Tagging with Implicit Feedback

Mohammad S. Sorower
Oregon State University
Corvallis, OR, USA
sorower@eecs.oregonstate.edu

Michael Slater
Oregon State University
Corvallis, OR, USA
slater@eecs.oregonstate.edu

Thomas G. Dietterich
Oregon State University
Corvallis, OR, USA
tgd@oregonstate.edu

ABSTRACT

Tagging email is an important tactic for managing information overload. Machine learning methods can help the user with this task by predicting tags for incoming email messages. The natural user interface displays the predicted tags on the email message, and the user doesn't need to do anything unless those predictions are wrong (in which case, the user can delete the incorrect tags and add the missing tags). From a machine learning perspective, this means that the learning algorithm never receives confirmation that its predictions are correct—it only receives feedback when it makes a mistake. This can lead to slower learning, particularly when the predictions were not very confident, and hence, the learning algorithm would benefit from positive feedback. One could assume that if the user never changes any tag, then the predictions are correct, but users sometimes forget to correct the tags, presumably because they are focused on the content of the email messages and fail to notice incorrect and missing tags. The aim of this paper is to determine whether implicit feedback can provide useful additional training examples to the email prediction subsystem of TaskTracer, known as EP2 (Email Predictor 2). Our hypothesis is that the more time a user spends working on an email message, the more likely it is that the user will notice tag errors and correct them. If no corrections are made, then perhaps it is safe for the learning system to treat the predicted tags as being correct and train accordingly. This paper proposes three algorithms (and two baselines) for incorporating implicit feedback into the EP2 tag predictor. These algorithms are then evaluated using email interaction and tag correction events collected from 14 user-study participants as they performed email-directed tasks while using TaskTracer EP2. The results show that implicit feedback produces important increases in training feedback, and hence, significant reductions in subsequent prediction errors despite the fact that the implicit feedback is not perfect. We conclude that implicit feedback mechanisms can provide a useful performance boost for email tagging systems.

Author Keywords

implicit feedback; TaskTracer; email tagging.

UIST '15, November 08-11, 2015, Charlotte, NC, USA.

http://dx.doi.org/10.1145/2807442.2807501

ACM Classification Keywords

H.1.m User/Machine Systems: Miscellaneous; I.4.9 Applications; I.2.6 Learning (K.3.2)

INTRODUCTION

Email was originally designed as a communications application. Over time, it has become a *habitat* for collaboration, a tool for time and task management, a medium for conversations and file transmission, and a mechanism for managing professional and social contacts for knowledge workers [9, 11, 21]. Workflows in organizations are often initiated, discussed, managed, and concluded via email exchanges [15]. *Whittaker et al.* [21] call this *email overload*, and show how the email inbox is employed as a repository for task to-do, to-read, and for tasks or correspondence in progress. However, with the increased volume of email that people receive every day, congested, unstructured and overloaded email boxes with disordered and unprioritized emails pose a critical bottleneck on efficient knowledge work. Tools are needed that can help the user efficiently organize and manage email, so that knowledge workers can keep pace with the stream of tasks. One of the most common practices among email users is to organize email messages manually into folders. This manual process clearly is not efficient and exhibits the same problems as folder-based file systems. There is no easy way to manage cases where an email message is related to multiple tasks or projects. Most email clients support hand-coded rules for tagging or foldering, but these must be manually crafted and they are mostly inflexible [6]. Therefore, recent attention has been drawn to applying machine learning methods to automatically classify email messages into folders or attach appropriate tags to the messages.

A substantial body of research has explored text-classification-based batch strategies for email classification [4, 17, 13, 1]. However, in the context of deployed email systems, batch methods are not appropriate. Instead, a good email tagging system should take the form of an online, multi-label classifier, and the classifier should be able to adapt quickly in response to user feedback. Furthermore, the classifier should be able to incorporate changes in the tag space and in the distribution of words in the email messages.

Two existing studies have explored online learning methods for email. First, *SwiftFile* [18] presented an incremental learning algorithm that predicts the three most-likely destination folders for an incoming message. This incremental learning method performed better than a periodic learning system (e.g., the classifier is updated after every thirty messages or overnight), because the periodic learning sys-

tem fails to promptly respond to user corrections. Second, an empirical evaluation of six different online learners for email classification was reported by *Keiser et al.* [12]. These online classifiers, namely, Bernoulli Naïve Bayes, Multinomial NaïveBayes, Transformed Weight-Normalized Complete Naïve Bayes, Term Frequency-Inverse Document Frequency Counts, Online Passive Aggressive, and Confidence Weighted, were tested using real email collected and tagged using the TaskTracer system [7]. The confidence weighted linear classifier [8] consistently performed better than all of the others.

TaskTracer Email Predictor 2 (EP2) [19] incorporates automated email prediction into Microsoft Outlook using a multilabel classifier based on the confidence weighted linear classifier. The user interface, which we will describe below, displays the predicted tags on each incoming message. The nice aspect of this is that if the tags are correct, the user doesn't need to take any action. In our experience, the classifier is 80-90% correct, and hence the cost to the user of manually tagging email messages is reduced by 80-90%. If the tags are incorrect, then the user can easily delete incorrect tags and add the correct tags, and this provides training examples to the classifier. However, the classifier receives no feedback when the predicted tags are correct. If the classifier was highly confident of those predictions, then this is not a problem. (Indeed, the confidence weighted classifier ignores positive feedback on confident predictions.) But if the classifier is not confident, then positive feedback confirming the correctness of those predictions would be very helpful.

The obvious response to this quandry is to employ what is known as "self training". The classifier could simply assume that its predictions are correct if the user doesn't make any corrections. However, our personal experience as users of EP2 has shown that once the classifier becomes reasonably accurate, users occasionally fail to provide corrective feedback, especially if they are working quickly or are deeply engaged in a task. Under such conditions, self training is risky.

This motivates us to explore *implicit feedback*. Our hypothesis is that the more time a user spends working on an email message, the more likely it is that the user will notice tag errors and correct them. The survival curve shown in Figure 1 supports this hypothesis. The horizontal axis shows the number of times (k) the user interacted with a message that had an incorrectly-predicted tag, and the vertical axis shows what fraction of the messages (with bad tags) survived k interactions. The survival curve shows an initial drop for $k = 1$ and $k = 2$ followed by slower decrease thereafter. This suggests that most of the bad tags are corrected almost immediately within a few interactions. Therefore, if the user spends a long time reading a message, saves an attachment, forwards the message to someone else, and then replies to the message—all without changing any of the tags—then it is likely the tags are correct. In contrast, if the user only briefly looks at the email message and then moves on to the next message, then the tags could very well be wrong and the user failed to notice. In this paper, we study implicit feedback mechanisms that record events in which the user interacts with an email

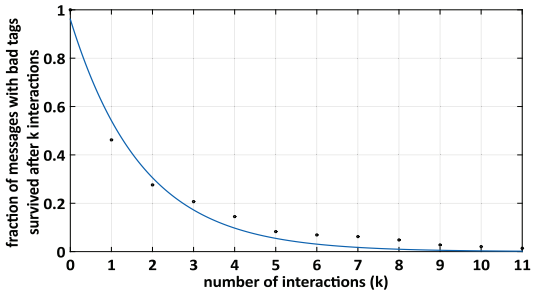


Figure 1: Survival curve (exponential fit to the data) showing the fraction of the messages with bad tags that survived k interactions with the user

message. If the total number of these events exceeds a threshold without any tag changes, then the tags are assumed to be correct and the classifier trains on that message.

Similar methods have been developed in information retrieval and recommendation systems. Those systems seek to determine whether the user found an item to be interesting versus uninteresting. The underlying model is that the user will spend more time reading documents that are more interesting among all the documents he/she is presented with. *InfoScope* [20] exploits this idea to learn user preferences for *Usenet* discussion group messages by combining explicit user-feedback with a few sources of implicit feedback such as whether the user read the message or not, whether the user saved the message or deleted it, whether the user followed-up on the message or not. *Morita et al.* conducted a user study over a six-week period with eight participants and showed that, just by monitoring *reading time*, it is possible to predict the user's degree of interest in particular Usenet articles [16]. A series of user studies, conducted in different domains, confirmed that user actions such as printing, forwarding, and scrolling a page have a strong positive correlation with the user's stated level of interest in the content of the presented article [14, 3, 10].

Although most of the work on implicit feedback research has been devoted to improving web-based applications, a recent realization among the research community is that the same approaches can contribute to developing smart desktop systems [2]. *Chirita et al.* propose a system that clusters desktop documents based on access timestamps, the number of steps between consecutive accesses of different files, and the time window they are accessed within [2]. The goal is to exploit the clusters to define desktop usage contexts and suggest context-related documents to the user.

This paper is organized as follows. We begin by describing the TaskTracer Email Predictor 2 (EP2) system and its user interface. We then describe the instrumentation that we added to EP2 to collect events relevant to producing implicit feedback. The next section describes three algorithms for generating implicit feedback in response to these events as well as two baseline methods. We then present a user study that collected events in which users are interacting with EP2-tagged email messages while they carry out various tasks specified

by those messages. The users were also asked to correct the tags. To evaluate and compare the various implicit feedback algorithms, we reprocess the data from the user study to create simulated users who notice and repair incorrect tags with a specified target probability. We then present and discuss the results of this algorithm comparison. The results show that by training on tags confirmed by implicit feedback, we can significantly improve the performance of the EP2 email classification system.

THE TASKTRACER EMAIL PREDICTOR EP2

Email Predictor 2 is a combination of a Microsoft Outlook add-in and a Java backend server. The Outlook add-in provides the user interface for performing tag operations, and it passes the data to the server to perform learning and prediction. Users define their own tag sets. The number of tags for long-term users of the system ranges from 50 to 350. To perform multi-label prediction, EP2 learns one CW linear classifier per tag. An email message with 3 user-assigned tags creates positive examples for 3 of these classifiers and negative examples for all of the others. We chose the CW classifier because of its aggressive update behavior. When the user corrects a classifier mistake on an email message, the classifier makes an "aggressive" update so that the email message would have been correctly classified with a large margin. This makes the classifier very responsive to user feedback, which is important to the user experience. A drawback is that the classifier is very sensitive to mislabeled training data. If the user flubs, for example by deleting the wrong tag or mistyping a tag, then the classifier will make a big change in its parameters. To address this, EP2 contains an automatic undo facility, so that if the user immediately corrects the flub, EP2 detects this and unwinds the parameter change [19].

When an email message arrives, it is processed to extract a binary feature vector. Features include information about the sender, the set of recipients, the words in the subject line, and words in the email body (after removing HTML markup and stopwords and performing stemming). The total number of features grows over time as new words and new email addresses are encountered. A typical feature vector contains 30,000 - 50,000 potential features of which only 30 - 200 are "turned on". The feature vector is then passed to each of the tag classifiers, which each produce a predicted probability that the email message should be assigned their tag. All tags with predicted probability above 0.7 are added to the email message and displayed to the user. If there are no such confidently-predicted tags, then the tag with the highest probability is "promoted" and added to the message. However, if all tags have predicted probability below 0.01, then no tags are promoted. Implicit feedback will be most useful if it can provide confirmation that the promoted tags are indeed correct.

Figure 2 shows the main components of the EP2 user interface. The tags assigned to a message are shown on the TaskBar (#1 in Figure 2) as click buttons (#3 in Figure 2). The user can easily remove a tag by clicking on the left side (on the red cross) of the button. To add a tag, the user has several options. First, the drop-down "combo" search box (#4 in Figure 2) allows the user to type the name of a tag. As the user types, matching tags appear in a drop-down menu, and the user can use the mouse or arrow keys to select the desired tag. For users with few tags, it is typically more convenient to click on the drop-down arrow and select the desired tag from the menu, as the menu is large enough to show at least 15 tags. A second approach is to use the drop-down menu near the "plus" sign (#5 in Figure 2). This dropdown provides two ways to add a tag. The lower part of the dropdown provides a menu of the 12 most-recently used (MRU) tags. The upper part of the dropdown provides a menu of the top five tags whose confidence was below the 0.7 prediction threshold. The user can add tags by clicking on entriees from either menu. These features are most helpful for users with many tags. The user can also create a new tag by clicking on the "+" button and typing the name of a tag.

The UI contains a few other components. A small control box (#2 in Figure 2) allows developers to stop and start the EP2 backend server. The user can also request updated predictions by selecting one or more email messages, right-clicking, and choosing "Predict tags for this message".

EP2 Instrumentation

To support implicit feedback, we added instrumentation to EP2 to capture and record information about the user's interaction with email messages in Microsoft Outlook. For each message, we computed the total number of times each of the following events occurred:

- message was opened and read in either the Outlook Explorer or the Outlook Inspector
- user added or removed a tag on the message
- user added or removed a flag from the message
- user moved the message to a folder
- user copied, replied, forwarded, or printed a message
- user saved an attachment from the message

We will refer to these events collectively as "implicit feedback events". Some of them require additional explanation as follows. The Outlook Explorer corresponds to the user interface shown in Figure 2, which displays a short summary of each email message and provides a "viewing pane" that displays the currently-select message. The Outlook Inspector displays a single email message in a separate window. Some users prefer the Explorer, others prefer the Inspector. Outlook allows the user to set various flags on a message such as "follow up today", "follow up tomorrow", "completed", and so on. Our instrumentation also tracks the total reading time for each message (i.e., the total time the window containing the message is in focus). But we did not use this information in our study, because without an eye-tracker, we do not know for a message window that is in focus, what fraction of the user's awareness has been spent on looking at the message or its tags, if any.

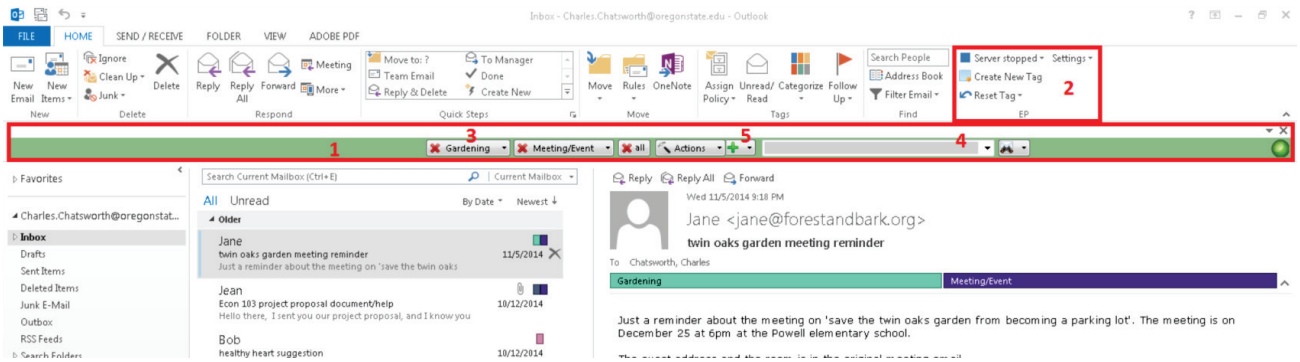


Figure 2: TaskTracer (Email Predictor) Tag Interface on Microsoft Outlook

ALGORITHMS

We designed three implicit feedback algorithms and two baseline algorithms for comparison. This section describes these algorithms. Each of these algorithms is designed to be invoked every time a new implicit feedback event occurs. The algorithm then decides whether enough implicit feedback has been received to conclude that the current set of tags on the message is correct. If so, it creates positive training examples for each tag on the message and negative training examples for each tag that is not on the message. This training is invisible to the user, and in fact, if the user subsequently changes a tag, the automatic undo facility will roll back the implicit training and train on the explicit feedback provided by the user.

No Implicit Feedback (NoIF).
This algorithm never creates implicit training examples. It only creates a training example for a tag if the user adds or removes that tag from the email message. This is the standard behavior of EP2, and it is a baseline against which to compare the other algorithms.

Simple Implicit Feedback (SIF).
When the user changes any tag (by deleting or adding a tag), the SIF algorithm immediately treats all remaining tags as correct and creates implicit feedback training examples (positive examples for the tags that are present and negative examples for all tags that are absent). The rationale for this is that if the user makes any changes to the tags, then the user has attended to the tags. Hence, any tags that the user does not change are highly likely to be correct. Note that if the user makes any subsequent tag changes, these are correctly handled by the undo system, as described above.

Implicit Feedback without SIF (IFwoSIF).
This algorithm maintains a count of the total number of implicit feedback events. It treats tag changes just like all other implicit feedback events. When this count exceeds a specified threshold, then it creates the implicit feedback training examples.

Implicit Feedback with SIF (IFwSIF).
This algorithm combines SIF and IFwoSIF. If the user changes a tag, then implicit feedback examples are immediately created. Otherwise, it continues to count up events until the number of implicit feedback events exceeds a specified threshold, at which point it creates the implicit feedback examples.

Online.
This algorithm mimics the online prediction framework studied in theoretical analysis of machine learning algorithms. The algorithm ignores all implicit feedback events. Instead, immediately after the tags are predicted for a message, the algorithm replaces the predicted tags with the correct tags and creates training examples for them. Hence, it provides perfect feedback to the EP2 multi-label classifier.

We measure the performance of each algorithm in terms of the total number of tag prediction errors. The *Online* baseline method should give the best results, because it provides ideal feedback, while the *NoIF* method should give the worst results, because it provides no implicit feedback. The central question of this paper is how well can the implicit feedback algorithms close the gap between *NoIF* and *Online*.

METHODS

We now describe the methods that we employed to answer this question. We first constructed and tagged a set of email messages. We then conducted a user study to collect user interaction data. Our goal in the study was to simultaneously encourage the study participants to correct the email tags while also engaging them in performing email-directed tasks so that they would frequently fail to notice incorrect predicted tags. Using the interaction data recorded from the participants, we then compared the implicit feedback algorithms by replaying the user interactions while manipulating the fraction of tags corrected by each participant.

Dataset of Tagged Email Messages

We created an email dataset from a variety of web sources. The messages in this dataset were chosen to reflect the email life of a knowledge worker—a student in this case. In our

scenario, the student is enrolled in two courses, is actively involved in projects, regularly attends meetings and events, and also has some hobbies. The dataset contains 330 messages with some of the messages also having file attachments. Based on its content, each message was tagged with one or more tags from a set of six possible tags: Economics, Entertainment, Gardening, Health, Math and Meeting/Event. Ground truth tagging was performed by two of the authors and one graduate student. Conflicts were resolved by taking a majority vote. The average number of ground truth tags per message was 1.24. Table 1 shows the distribution of tags. The messages are very similar to what a typical student would receive in everyday life. Some of the messages are completely informational (e.g., a professor describing a homework assignment). Others contain a request asking the recipient to perform one of the following tasks: save the attachment(s) from the message, edit a saved file and attach it to a reply or a new out-going email message, send (reply or forward) messages with or without attachments, find requested information on the web, copy it into an email message, and send it. Here is an example email message:

```
Sender: Bishop <bishop@appqualify.com>
Subject: help! math midterm solution

hi,
Can you please FORWARD me the midterm
solution the professor sent a few days
back? Somehow I don't find that email!

thanks
-Bishop
```

Notice that the task is cued with capital letters so that the study participants can easily recognize it.

After ground-truth tagging, the messages were randomly divided into four sets as follows. (All sampling was stratified to ensure that the frequency of the tags was approximately balanced within each set.) First, the 330 messages were divided into a set ("Train60") of 60 messages for training and a set ("Test270") of 270 messages for testing. The Test270 set was further divided into three sets: "Task1" with 66 messages, "Task2" with 102 messages, and "Task3" with 102 messages. The mean number of tags per message was 1.27 for Train60 and 1.23 for Test270.

To prepare the messages for the user study, we trained EP2 on the ground truth tags for the messages in "Train60". We then applied the learned multi-label classifier to predict tags on all of the "Test270" messages. After training on only 60 messages, the classifier is not very accurate. Consequently, the predicted tags contain many errors. This was intentional, because we wanted to give the participants many incorrect tags to correct.

Lab-controlled User Study

We conducted a lab-controlled user study with a total of 15 participants. Only adult email users who receive 20 or more email messages everyday and who regularly use tags, categories, labels, or folders to organize email, were recruited to participate in the study. We also collected information about the participants' current email usage and email organization methods. On average, the participants received 37 emails per day. 87% of the participants regularly use Gmail or Google Apps as their primary email client, and the remaining 13% use Microsoft Outlook. 80% of the participants had employed at least some tags, categories, labels, or folders to organize their messages in the past two weekdays. Most of the participants (71%) regularly use labels or folders. They organize their email using some combination of manually-created rules, interactively assigning labels, and interactively moving messages to folders. About 50% of the participants regularly use tags or categories (the tags provided natively in Outlook). Here are a few comments from the participants about their email organization methods:

Tags	%messages
Economics	15
Entertainment	18
Gardening	19
Health	23
Math	17
Meeting/Event	31

Table 1: The distribution of tags in the email dataset. For each tag, this table shows the percentage of messages that were assigned that tag. This totals to more than 100% because a message may have multiple tags.

"I transfer to a particular folder then work later."

"I have created different folders in my mail like Research, personal, work etc. Depending on the type of email I receive, I label and move the particular mail to respective folder. For example if I receive any email regarding internship, I will move it to folder work, so that it will be easily accessible to me whenever required. I will do labeling, moving, tagging using the options which we get in gmail(labels, move to etc)."

The study participants interacted with Microsoft Windows and Outlook via a remote virtual terminal connected to a Windows server. This allowed us to completely control the desktop environment during the study so that the participants were not interrupted by their regular email flow, chat windows, calendar notifications, and so on.

The study was conducted in three two-hour sessions on three separate days. The first session was divided in half. During the first half, the students were asked to use the EP2 UI to tag messages that had been preloaded into their inbox. These are the "Train60" messages, and they were presented without any tags. While adding appropriate tags on these messages, the users learned the intended meanings of the tags and the properties of the email messages. They also learned how to carry out the tasks requested by the messages (e.g., how to save attachments, add attachments, reply, forward, etc.). Most importantly, the participants learned about the student role they were playing in the study.

In the second half of the first session, the participants were presented with the "Task1" messages in the inbox along with

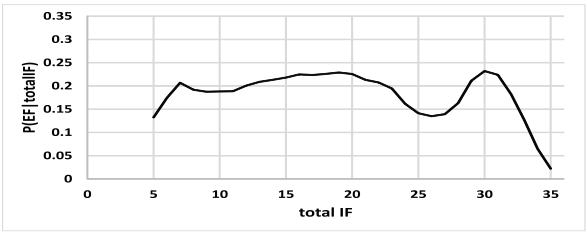


Figure 3: Conditional probability distribution, *P(EF| totalIF)*.

the (error-prone) predicted tags. During this session, the participants were told to perform the tasks that are described in the email messages. In addition, they were also asked to correct any tags that they notice are incorrect.

In the second and third sessions, the participants were asked to work on the "Task2" and "Task3" messages, respectively.

At the end of each session, the participants filled out a Qualtrics questionnaire that required them to provide the tags they believed were correct for each email message. We will call these the "user ground truth" tags.

Out of 15 participants, one participant dropped out during the course of the study. Therefore, we only consider for subsequent analysis the data from the 14 participants who successfully completed all three sessions.

Post-study Simulation

When a participant adds or deletes a tag from an email message, we will say that the participant has provided "Explicit Feedback" (EF). An initial analysis of the data collected from the study showed that the participants did not provide very much explicit feedback. The mean percentage of messages for which they corrected tags was 16.3% (standard deviation 0.9%). This is much lower than we have observed for long-term users of TaskTracer EP2, where it varies between 60% and 90%. This shows that the participants focused primarily on performing the tasks specified by the email messages and paid less attention to our request that they fix incorrect tags. With such low levels of explicit feedback (and with many incorrect tags), interpreting implicit feedback becomes very difficult.

We addressed this shortcoming by combining the observed implicit feedback events with simulated explicit feedback as follows. For each participant and each email message, let the variable EF be 1 if the participant provided explicit feedback and 0 otherwise. Similarly, let the variable $totalIF$ denote the total number of implicit feedback events observed for the participant on that email messages. From the set of observed $(EF, totalIF)$ pairs, we can estimate the conditional probability $P(EF = 1|totalIF)$ that a randomly-selected participant will provide explicit feedback on a randomly-chosen email message given the number of implicit feedback events that they have produced. A smoothed version of this estimated distribution is shown in Figure 3. Note that the probability of providing EF is fairly constant (at around 0.2) as a function of the amount of IF, but drops off for very high levels of IF.

We then designed and implemented Algorithm 1 to modify the explicit feedback data $\mathcal{D}^p$ collected for participant p to achieve a target level tEF of explicit feedback. The algorithm begins by constructing a vector of predicted explicit feedback probabilities $\overline{pEF}$ based on the estimated distribution $P(EF|totalIF)$ for participant p. Then it compares the fraction of observed EF to the target tEF and computes the number of messages n whose EF must be changed. If the observed EF is too low, then explicit feedback is added to the n messages with highest predicted probability of EF. If the observed EF is too high, then explicit feedback is removed from the n messages with lowest predicted probability of EF.

Algorithm 1 SampleEF(p, tEF)

Input: p = User id for the participant,
$\mathcal{D}^p$ = User actions for messages for participant p,
$P(EF|totalIF)$ = fitted probability of EF given total IF,
tEF = target level of EF
Output: $\mathcal{D}^p_{tEF}$ = User actions for participant p achieving tEF

$\overline{EF} \leftarrow$ vector of observed EF for all messages
$\overline{IF} \leftarrow$ vector of observed IF for all messages
$\overline{pEF} \leftarrow$ vector containing $P(\overline{EF}|\overline{totalIF})$ for messages
$N \leftarrow$ total # of messages for p
$EF \leftarrow$ # observed messages with explicit feedback for p
$ObservedEF \leftarrow \frac{EF}{N}$ observed probability of EF for p
$n \leftarrow |ObservedEF - tEF)| \cdot N$
    number of messages to change
if $ObservedEF > tEF$ **then**
    $M \leftarrow n$ messages in increasing order of $\overline{pEF}$
    $\mathcal{D}^p_{tEF} = \mathcal{D}^p$, after removing EF from messages in M
if $ObservedEF < tEF$ **then**
    $M \leftarrow n$ messages in decreasing order of $\overline{pEF}$
    $\mathcal{D}^p_{tEF} = \mathcal{D}^p$, after addingEF to messages in M
Return $\mathcal{D}^p_{tEF}$

Algorithm 1 was applied to generate simulated event streams for each participant for tEF values of 0.2, 0.3, 0.4, 0.5, 0.7, and 0.8. These were then processed by EP2 as follows. First, EP2 was trained on the ground truth messages in "Train60". Then the simulated events were processed by EP2 via a special "replay mode" using each of the five implicit feedback algorithms. To assess performance, we measured the cumulative number and fraction of tags incorrectly predicted.

The IFwSIF and IFwoSIF algorithms employ a threshold to decide when to create training examples. To set that threshold, we randomly selected a few of the participants and examined their simulated event data for different levels of target EF. We evaluated a few thresholds for each stream and took a majority vote to select the best threshold. We then use this best threshold (a total of 7 IF events) for all our experiments. There was strong agreement on the range of good settings across participants and target EF values.

RESULTS

Figure 5 summarizes the implicit feedback events collected from the study participants, summed over the three sessions.

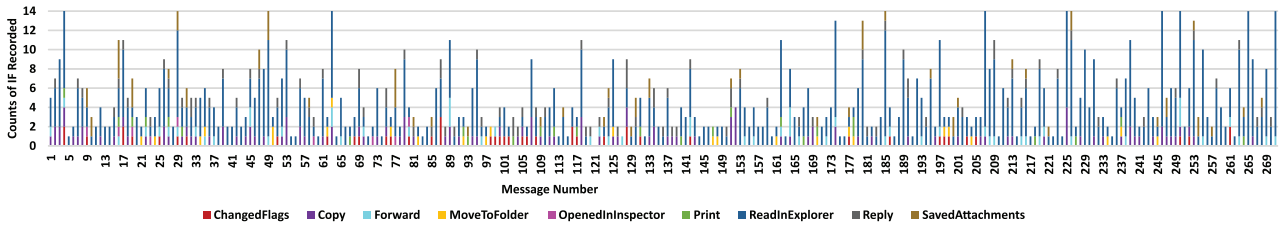


Figure 4: Implicit feedback captured during the study sessions of one participant. The first session ends after message 66, and the second session ends after message 168.

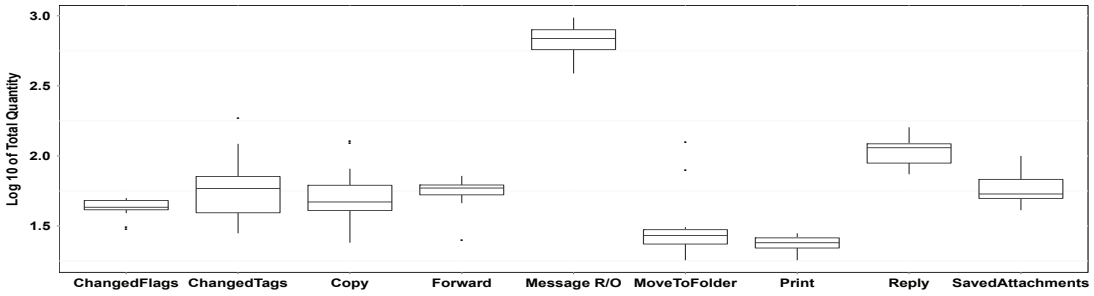


Figure 5: Total number of implicit feedback events captured (log scale) for each type of implicit feedback event. 'Message R/O' indicates the total number of times the message was opened in Outlook Explorer or in Outlook Inspector.

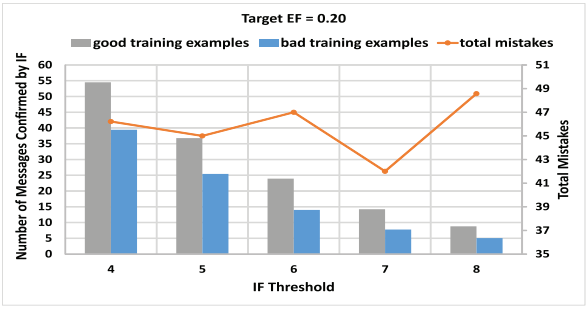


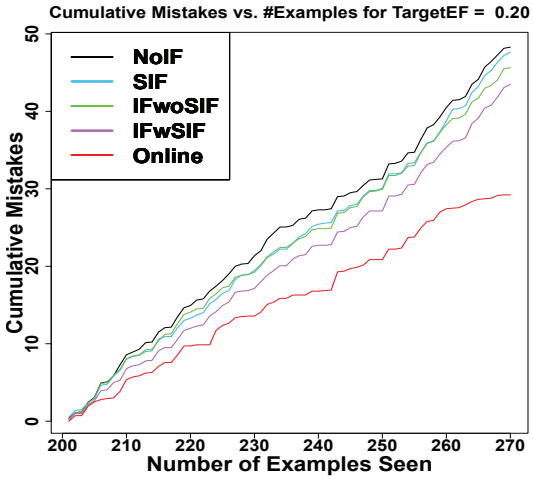
Figure 6: Total mistakes (right axis), total number of good and bad training examples (left axis) created by IFwSIF for different levels of the implicit feedback threshold (TargetEF = 0.20)

We employed a log scale in order to fit the wide range of values within a single plot. The narrow inter-quartile ranges of the box plots show that the distribution of these values is quite similar across the different participants. This reflects the fact that each participant was given the same set of email messages with the same tasks. The largest variation is observed in the number of tags changed and the number of times the email messages were opened (although the box for the latter appears small in the figure because of the log scale).

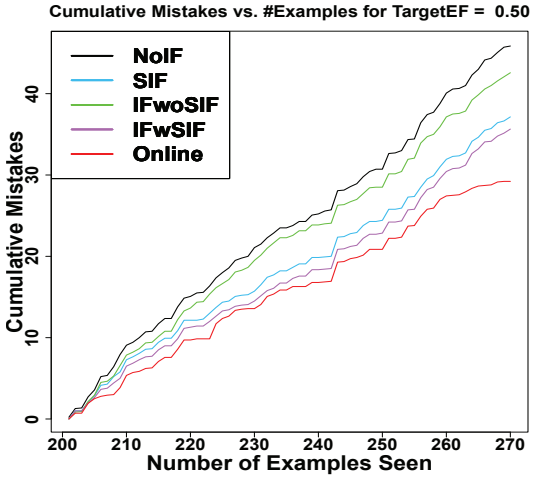
Figure 4 plots a timeline of the implicit feedback events for one study participant. We can see that the implicit feedback events are evenly distributed throughout the three study sessions.

As discussed above, without IF, the only machine learning choices are to train on all predictions or to train only on the EF. If we train on all predictions, the incorrectly-predicted tags will create bad training examples. If we train only on EF, we get fewer training examples, but they are all correct. An IF strategy seeks an intermediate path. It will succeed if there is a threshold on the number of IF actions such that the loss in accuracy of the resulting incorrect training examples (created from "surviving" bad tags) is out-weighed by the gain of the resulting correct training examples. To evaluate this, we plot the number of bad and good training examples, and the final cumulative number of mistakes as a function of the threshold (as shown in Figure 6 for target EF level of 0.20). Notice that the final cumulative number of mistakes in Figure 6 has a minimum (at IF threshold = 7.0).

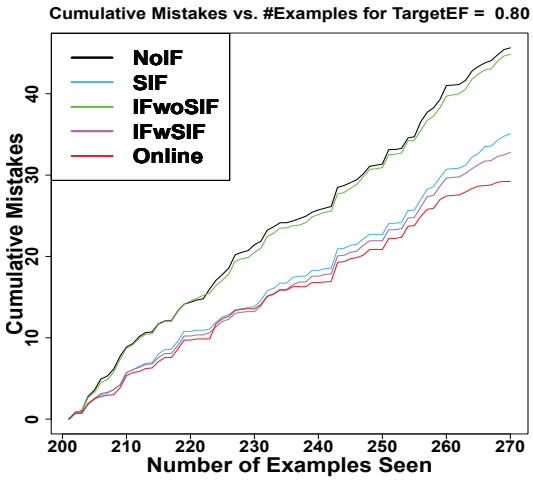
Let us now consider the core question: Do the various implicit feedback algorithms improve classifier accuracy? We do not expect implicit feedback to provide much gain early in the experiment, because the predicted tags are not very accurate. As our goal is to assess the effectiveness of implicit feedback for long-term users of EP2, we focus our analysis on the final 70 email messages. Figure 7 plots the cumulative prediction mistakes (averaged across all participants) of each of the five algorithms for three target EF levels: 0.20, 0.50, and 0.80. A target EF of 0.20 is close to the actual behavior of the participants in the experiment. A target EF of 0.80 is typical of behavior exhibited by long-term users of EP2, and a target EF of 0.50 is plotted to show an intermediate level of explicit feedback. In all cases, our baseline methods NoIF (no implicit feedback) and Online (complete



(a) $tEF = 0.20$



(b) $tEF = 0.50$



(c) $tEF = 0.80$

Figure 7: A comparison of the cumulative mistakes of each of the five IF algorithms on the last 70 email messages for three values of TargetEF

Alg. \ tEF	0.20	0.30	0.40	0.50	0.70	0.80
NoIF	11.43	11.67	10.48	10.95	10.95	10.95
SIF	11.43	10.00	8.81	8.81	8.33	8.33
IFwoSIF	10.86	10.19	9.86	10.13	10.92	10.68
IFwSIF	10.48	10.00	8.57	8.57	7.86	7.86
Online	6.90	6.90	6.90	6.90	6.90	6.90

Table 2: Percentage tag prediction mistakes

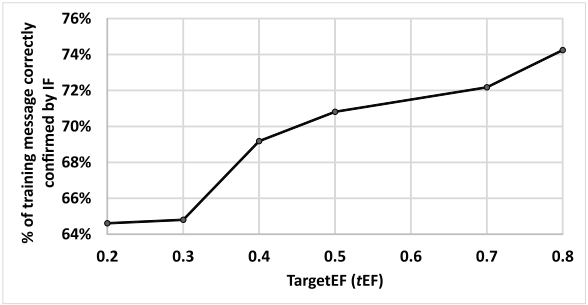


Figure 9: Percentage of training messages correctly confirmed by IF for different levels of target EF

online feedback) accumulate the largest and smallest number of errors, as expected. For target EF levels of 0.50 or more, IFwSIF produces the fewest errors, SIF is second best, and IFwoSIF is the worst. This shows that simple implicit feedback (i.e., training as soon as the user changes any one tag) and implicit feedback (i.e., training when the number of implicit feedback events exceeds a threshold) both provide good training examples. Combining them using IFwSIF gives better results than either method alone. For a target EF of 0.20, the implicit feedback algorithms do not produce very large error reductions compared to NoIF. But for a target EF of 0.80, the SIF and IFwSIF algorithms have largely eliminated the gap between NoIF and Online.

Table 2 provides another view of this information. It reports the percentage of tag prediction mistakes on the last 70 email messages for each of the target levels of EF. Here we can see quantitatively that at a target EF of 0.20, the implicit feedback methods are giving only a small benefit over NoIF. But as the target EF level rises, 75% of the gap between NoIF and Online has been eliminated by IFwSIF.

TargetEF is the fraction of messages where the simulated user will examine and correct tags. As TargetEF increases, the classifiers become more accurate and make fewer mistakes, which reduces the number of messages that require EF. Figure 8 plots the number of training examples (on the entire dataset) that the NoIF, SIF, and IFwSIF methods generate. Observe that SIF produces the predominant share of the training examples. Nonetheless, the additional examples added by implicit feedback have a substantial effect on further reducing prediction errors. This is indirect evidence that those examples are accurate. Using IFwSIF, the classifier receives 64%

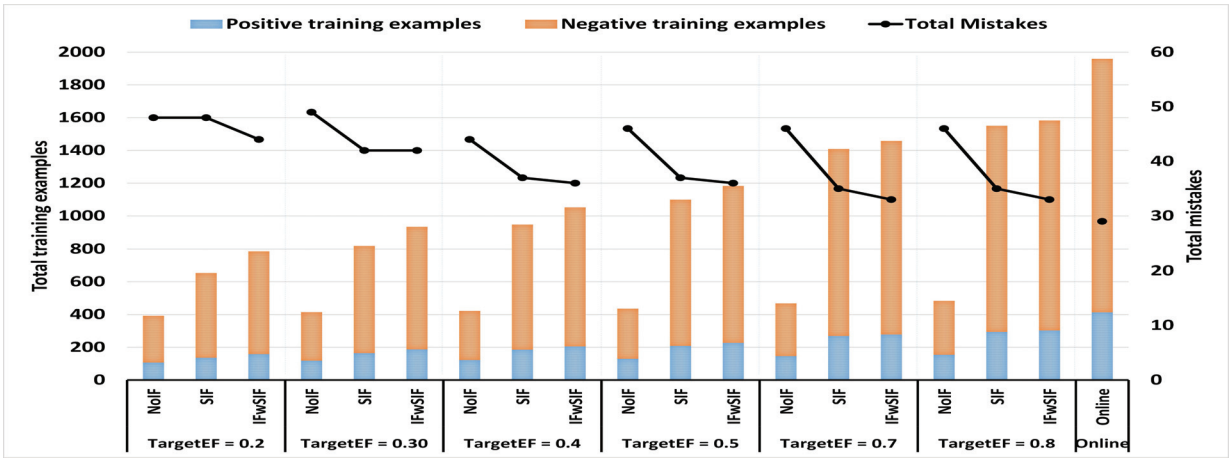


Figure 8: Total number of training examples for the entire experiment (left axis) and total number of prediction mistakes on the last 70 messages (right axis) for different levels of targetEF.

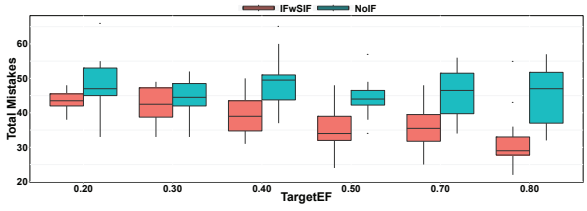


Figure 10: Total mistakes for different levels of targetEF. p-value < 0.05 for a two-sided Welch's two sample t-test ensures that we have sufficient evidence to conclude that the total number of mistakes in $NoIF$ is greater than the number of mistakes in $IFwSIF$.

more training than NoIF and 14% more training than SIF (averaging across all levels of target EF).

Figure 9 provides additional insight into this issue of the quality of the implicitly-confirmed training examples. It reports the percentage of the confirmed email messages that were correctly confirmed by IFwSIF for various levels of target EF. We see that in all cases, implicit feedback is doing better than random. However, at a Target EF of 0.20, only 64% of the confirmed messages have correct tags, whereas at a Target EF of 0.80 this has improved to 74%. In all cases, this shows that the training examples created by IFwSIF contain a lot of noise in the target tags. We were pleasantly surprised to see that on balance the classifier still benefited from these noisy training examples. This was particularly surprising because of the well-known vulnerability of the confidence weighted classifier to mislabeled training examples.

Finally, to test the statistical significance of the error reduction, Figure 10 displays box plots of the total mistakes on the last 70 messages for NoIF and IFwSIF. At each level of TargetEF and for all levels combined, the differences between the two algorithms are statistically significant at $p < 0.05$. This provides strong evidence that IFwSIF is giving a real improvement over NoIF.

DISCUSSION

When we designed the user study, we hoped that we could induce the participants to fix about 50% of the incorrect tags. We were disappointed when their explicit feedback rates were less than 20%. Nonetheless, we believe that our post-study simulation provides a good approximation to the behavior of long-term users. The explicit feedback that is added in the simulation is based on the observed EF behavior of the participants and does not build in our prior belief that more extensive interaction with an email message should be correlated with increased probability of providing explicit feedback.

The results of the experiment are very encouraging, and they suggest that there is additional room to improve the effectiveness of implicit feedback. First, our current approach treats all forms of implicit feedback events as being equally informative. With a larger sample, and perhaps using eye-tracking, we could determine which IF events are more likely to cause the user to notice incorrect tags. We could incorporate such information to compute weights on the IF events and then compare the weighted IF against a threshold. Second, in our study, the classifiers for each of the tags received approximately equal amounts of training. In real applications of EP2, some tags are much more common than others and so have many more positive training examples. In addition, new tags are introduced frequently, so that some classifiers have very few training examples. Of course the accuracy of a classifier increases as it is given more examples. Therefore, one might expect that the implicit feedback threshold might need to be higher for messages with poorly-trained tags and lower for well-trained tags. It would be interesting to test this hypothesis. Third, our study shows that the training examples created by IFwSIF still contain many incorrect tags. This suggests that we should either modify the confidence weighted clas-

sifier to make less aggressive updates when trained on these examples or else switch to an online learning algorithm that is more robust to label noise, such as AROW [5].

There are also potential improvements in the user interface that could encourage the user to provide more explicit feedback. For example, because feedback is most useful on tags for which the classifier has low confidence, we could provide a visual cue to the user (e.g., by changing the color of the tag button) to call attention to those low-confidence tags. We could add a drop-down option for the user to confirm that the tag is correct. If the user selected this, then the tag color could change back to a neutral or even a positive color.

Finally, tags could support functions beyond simple email organization. In the original TaskTracer system, when the user saved an attachment (or opened a file to attach to an email message), the "current task" of the user was consulted to suggest appropriate task-related folders in the open/save dialogue box. A similar idea could be implemented using tags. To save an attachment, the user could click on a tag to reveal "Save Attachment" drop-down menu item. The folders associated with that tag could be presented in the open/save dialogue. A similar interaction would make adding attachments to outgoing messages easier. The more functions that tags support, the more motivation the user has to fix incorrect tags. In the long run, this might reduce the need for an implicit feedback mechanism.

CONCLUSION AND FUTURE WORK

Any tagging system must cope with the problem of incomplete feedback, both because users will forget to provide feedback and because users will generally only provide corrective feedback rather than confirming that predicted tags are correct. The results of our investigation make it clear that an email tagging system can benefit from even a simple implicit feedback confirmation system. We studied two sources of implicit feedback. Simple Implicit Feedback (SIF) captures the case where the user corrects one tag on an email message. The remaining tags (absent or present) are thereby confirmed as being correct. Implicit Feedback without SIF (IFwoSIF) is based on the hypothesis that the longer a user interacts with an email message, the more likely they are to notice and correct any incorrect tags. Both of these sources of feedback were shown to provide substantial additional training examples to the classifier, which produced good reductions in prediction errors. Their combination, IFwSIF, provided even better error reductions. The error reductions were obtained despite the fact that the additional training examples produced by these implicit feedback mechanisms had a fairly high rate of erroneous tags. We therefore recommend that machine-learning-based tagging systems should incorporate implicit feedback mechanisms to extract more information from the user's interaction with the user interface.

ACKNOWLEDGMENTS

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. 2014-1451 (CFDA 12.XXX) via the Air Force Research Lab (AFRL) and administered by Galois Inc. of Portland, OR. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA, the Air Force Research Laboratory (AFRL), Galois or the US government.

REFERENCES

1. R. Bekkerman, A. McCallum, and G. Huang. 2004. Automatic Categorization of Email into Folders: Benchmark Experiments on Enron and SRI Corpora. *Center for Intelligent Information Retrieval, Technical Report IR* 418 (2004).
2. Paul-Alexandru Chirita, Stefania Costache, Julien Gaugaz, and Wolfgang Nejdl. 2006. Desktop Context Detection Using Implicit Feedback. *Personal Information Management: Now That Were Talking, What Are We Learning?* (2006), 24.
3. Mark Claypool, David Brown, Phong Le, and Makoto Waseda. 2001. Inferring User Interest. *IEEE Internet Computing* 5 (2001), 32–39.
4. W. W. Cohen. 1996. Learning Rules that classify E-mail, In AAAI Spring Symposium in Information Access. *AAAI Spring Symposium in Information Access* (1996).
5. Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive Regularization of Weight Vectors. In *Advances in Neural Information Processing Systems 22*. 414–422.
6. Elisabeth Crawford, Judy Kay, and Eric McCreath. 2002. IEMS - The Intelligent Email Sorter. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML '02)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 83–90. http://dl.acm.org/citation.cfm?id=645531.656016
7. Anton N. Dragunov, Thomas G. Dietterich, Kevin Johnsrude, Matthew Mclaughlin, Lida Li, and Jonathan L. Herlocker. 2005. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *In IUI 05: Proceedings of the 10th international conference on Intelligent user interfaces*. ACM Press, 75–82.
8. Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-Weighted Linear Classification. In *International Conference on Machine Learning (ICML)*.
9. Nicolas Ducheneaut and Victoria Bellotti. 2001. E-mail as habitat: an exploration of embedded personal information management. *Interactions* 8, 5 (Sept. 2001), 30–38. DOI: http://dx.doi.org/10.1145/382899.383305
10. Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.* 23, 2 (April 2005), 147–168. DOI: http://dx.doi.org/10.1145/1059981.1059982
11. Jacek Gwizdka and Mark H. Chignell. 2004. Individual differences and task-based user interface evaluation: a

case study of pending tasks in email. *Interacting with Computers* 16 (2004), 769–797. Issue 4. DOI: http://dx.doi.org/10.1016/j.intcom.2004.04.008

12. Victoria Keiser and Thomas G. Dietterich. 2009. Evaluating online text classification algorithms for email prediction in TaskTracer. In *Conference on Email and Anti-Spam*.

13. Svetlana Kiritchenko and Stan Matwin. 2001. Email Classification with Co-training. In *Proceedings of the 2001 Conference of the Centre for Advanced Studies on Collaborative Research (CASCON '01)*. IBM Press, 8–. http://dl.acm.org/citation.cfm?id=782096.782104

14. Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, John Riedl, and High Volume. 1997. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM* 40 (1997), 77–87.

15. Wendy E. Mackay. 1988. More than just a communication system: diversity in the use of electronic mail. In *Proceedings of the 1988 ACM conference on Computer-supported cooperative work (CSCW '88)*. ACM, New York, NY, USA, 344–353. DOI: http://dx.doi.org/10.1145/62266.62293

16. Masahiro Morita and Yoichi Shinoda. 1994. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag, New York, NY, USA, 272–281. http://dl.acm.org/citation.cfm?id=188490.188583

17. Jason D. M. Rennie. 2000. ifile: An Application of Machine Learning to E-Mail Filtering. In *Proc. KDD Workshop on Text Mining*.

18. Richard Segal and Jeffrey O. Kephart. 2000. Incremental Learning in SwiftFile. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 863–870. http://dl.acm.org/citation.cfm?id=645529.657947

19. Michael Slater, Thomas Dietterich, and Mohammad Sorower. in preparation. TAPE: An Integrated Machine Learning System for Email Tagging. (in preparation).

20. Frank Curtis Stevens. 1993. *Knowledge-based Assistance for Accessing Large, Poorly Structured Information Spaces*. Ph.D. Dissertation. Boulder, CO, USA. UMI Order No. GAX93-20482.

21. Steve Whittaker and Candace Sidner. 1996. Email overload: exploring personal information management of email. In *CHI 96 Conference On Human Factors In Computing Systems*. ACM Press, 276–283.