Agent-Assisted Task Management that Reduces Email Overload

Andrew Faulring, Brad Myers, Ken Mohnkern, Bradley Schmerl, Aaron Steinfeld, John Zimmerman, Asim Smailagic, Jeffery Hansen, and Daniel Siewiorek
School of Computer Science, Carnegie Mellon University
{faulring, bam, kem, schmerl, astein, johnz, asim, hansen, dps}@cs.cmu.edu

ABSTRACT
RADAR is a multiagent system with a mixed-initiative user interface designed to help office workers cope with email overload. RADAR agents observe experts to learn models of their strategies and then use the models to assist other people who are working on similar tasks. The agents' assistance helps a person to transition from the normal email-centric workflow to a more efficient task-centric workflow. The Email Classifier learns to identify tasks contained within emails and then inspects new emails for similar tasks. A novel task-management user interface displays the found tasks in a to-do list, which has integrated support for performing the tasks. The Multitask Coordination Assistant learns a model of the order in which experts perform tasks and then suggests a schedule to other people who are working on similar tasks. A novel Progress Bar displays the suggested schedule of incomplete tasks as well as the completed tasks. A large evaluation demonstrated that novice users confronted with an email overload test performed significantly better (a 37% better overall score with a factor of four fewer errors) when assisted by the RADAR agents.

Author Keywords
Agents, email classification, email overload, intelligent planning, learning, RADAR, task management.

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces---Interaction styles, Graphical user interfaces (GUI); I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence---Intelligent agents, Multiagent systems.

General Terms
Design, Experimentation, Human Factors.

INTRODUCTION
Email plays a central role in the work of many people. Unfortunately email client software is poorly suited to support the "collaborative quality of e-mail task and project management," which results in people suffering from "email overload" [2]. When faced with a large number of emails, people can find it difficult to choose an order in which to handle them. Possible strategies include handling each email in the order received, scanning the list of email subjects and senders for ones that appear to match various criteria (important, quick to handle, and so forth), or using filters to file emails into folders. The order in which emails are handled can significantly affect the efficiency of the strategy, since performing similar tasks together reduces the overhead of switching between different types of tasks. People find it difficult to create an efficient order when sorting their inbox using email-centric properties, such as sender, subject, or date, because sorting by those properties will usually not group similar tasks together nor will it account for inter-task dependencies. Email threads can help in some situations, and other times people can rely on "crutches" such as asking senders to use a specific subject line in the email. However in general, a person would need to inspect each email, manually create task metadata, and then generate an order for handling the emails. Several research projects have explored adding task-management features to email clients [2, 12, 23], and some email clients do provide features that facilitate task management such as tagging and separate to-do lists. The primary drawback of this approach is that users resist doing that additional work [24], and it forces them to read each email at least twice: once when creating the tasks, then again to actually do the task. Temporal factors further complicate email processing. High priority emails might need to be handled independent of efficiency concerns. Additionally, when time is tight people must decide which emails to ignore.

We have developed a mixed-initiative email system, which uses Artificial Intelligence (AI) learning techniques, to help mitigate these problems and thereby reduce email overload. We evaluated our system in an experiment in which RADAR was trained by observing people who were not the test participants. While this tests a situation where a novice is filling in for an expert, we are also interested in how well our techniques will help when a person is processing their own email. The Email Classifier observes the types of tasks an expert creates from emails and then learns a model that it uses to automatically find tasks within new emails. Next, the Multitask Coordination Assistant (MCA) observes expert users perform the tasks found by the Email Classifier. The MCA uses these training observations to learn models

IUI'10, February 7–10, 2010, Hong Kong, China.
Copyright 2010 ACM 978-1-60558-515-4/10/02...$10.00.

describing how to efficiently perform a set of tasks. The MCA uses these learned models to assist users working on a similar set of tasks. Its advice includes a suggested order for performing tasks and warnings when the user's behavior differs significantly from the expert's behavior. This approach reduces the number of times a person has to read an email and allows the tasks to be performed efficiently.

This mixed-initiative email system was a central feature of the **R**eflective **A**gents with **D**istributed **A**daptive **R**easoning (RADAR) project. RADAR was a large interdisciplinary project to build a suite of intelligent agents that help office workers complete tasks more efficiently. Over 100 faculty, staff, and students worked on RADAR components including the Email Classifier [25] and Multitask Coordination Assistant, along with other components including a Natural Language Processor [17], a Knowledge Base [7], a Schedule Optimizer [9], a Webmaster [27], a Briefing Assistant [15], and a Task Management Architecture [11]. Freed provides a more detailed description of the overall RADAR approach, architecture, and agents [10]. We have previously described some of the user interfaces in a case study on usability issues for AI systems [8]. The current paper provides more details on the interface design and a significantly more detailed analysis of the user study results.

A large-scale user test evaluated several versions of RADAR over three years [19]. The test measured RADAR's performance using quantitative metrics acquired through data logging, including an evaluation score that summarizes overall performance along with qualitative metrics collected with a post-test user survey [20].

The post-test user survey for the RADAR 1.1 system found that RADAR's AI had little impact on user perceptions of the system. Although RADAR 1.1 included the Email Classifier, tasks were embedded as objects in each email. The system also exhibited usability and performance problems. It was hypothesized that the poor user interface was masking the potential benefit of the AI [20]. For RADAR 2.0, we designed a mixed-initiative task-centric email client in which the user's inbox is augmented with a separate task list. The post-test survey for the RADAR 2.0 study found that the AI technologies now positively impacted user perceptions of RADAR. In particular, participants were more confident that they had done tasks well, had found tasks easier to complete, and had been more immersed in the test. These qualitative metrics were supported by a significant improvement in the test's evaluation score.

While participants were more successful at performing individual tasks, many still struggled to find an efficient order for performing a group of tasks. We observed a wide variance in performance including a long-tail of poor scores from participants who struggled. To address this problem, the RADAR 3.0 system introduced a task strategy component called the Multitask Coordination Assistant. The MCA proposes a schedule for performing tasks, emphasizes highly-important "critical" tasks, recommends tasks to skip if time is tight, and issues warnings if the user's behavior deviates significantly from expert behavior. The results of testing RADAR 3.0 showed another significant increase in performance accompanied by a noticeable drop in variance.

These results suggest that adding AI technologies to interactive systems can benefit users, a conclusion that has been met with skepticism within the HCI community [18]. While some of the concern focuses specifically on anthropomorphic agent interfaces, a major complaint against interactive AI is that it interferes with a user's ability to easily *predict*, *control*, and *monitor* the system's behavior. When the AI technologies make mistakes, will the user notice and correct the errors? When the system does act correctly, will the user notice what has been done? A related concern is that people will have difficulty *understanding* why an intelligent system made a particular decision or suggestion, which might lower a user's trust or confidence in the system. However, the benefits of predictability and understandability need to be weighed against the benefits of automation [13]. A direct manipulation system can be slower and have *more* errors than an intelligent system [21]. Predictability and understandability are subgoals of the ultimate goal, usable systems, and must be weighed against other subgoals such as performance, which also impact usability.

RADAR makes specific contributions to address these issues. The automatically detected tasks are presented in a list view and in a detail view along with the original email message, so that users can easily understand, check and perform the tasks. The recommended ordering of the tasks is clear from the sort order in the task view and in a novel Progress Bar that shows future as well as completed tasks. Additionally, RADAR is generally not allowed to take autonomous actions when such actions would be visible to other people or systems. RADAR has considerable flexibility to work on the user's behavior without risking costly mistakes that might embarrass the user or leak private information.

A user study showed that the user interfaces presented the AI assistance in a helpful manner: users who received AI assistance performed 37% better compared with users who did not. Additionally, users were able to recognize when RADAR made errors, correctly handling 89% of the tasks that the Email Classifier erroneously suggested. Overall, these users incorrectly completed 2.6 tasks per user. In contrast, the users without AI assistance *incorrectly* completed 10.3 tasks on average, which accounted for 19% of the tasks they did. This is a factor of four *more* errors.

RELATED WORK

Much task-management research has studied adding task-management features to email client software [2, 12, 23] and making it easy to move tasks from email client software to dedicated task managers [21, 23, 27]. This work focuses on email because users often use email client software as a task manager since many tasks arrive as emails and the information necessary to complete a task is commonly contained in the email as well. Hence, the inbox becomes an informal to-do list [6]. Unfortunately, email applications are not designed to perform the task-management duties that users demand from them [24]. For example, when an inbox collects a large number of emails, it can become difficult to find information. Email software allows users to create folders and email filters, label emails, perform searches, and so forth. Unfortunately these features are not always implemented in a way that makes task management easy [23]. For example, Gwizdka found that most of the Microsoft Outlook users he studied did not use many of Outlook's features, such as the to-do list, email flags, reminder flags, and journal [12]. Dedicated task-management applications do not provide the answer to this problem since users dislike the tedious process of entering task metadata such as dependencies, due dates, and priorities [1]. TaskMaster [2], which did not use AI, took a hybrid approach like RADAR, providing features of both email clients and task managers.

Task managers generally display to-do tasks in a list [1, 3, 16], as does RADAR. One notable exception is TaskView [12], which displays future tasks within an email client by arranging emails in a grid where time runs along the horizontal axis and an email property, such as sender or subject, runs along the vertical axis. However, TaskView requires that task information be manually entered, whereas RADAR calculates task priorities and orders tasks automatically in both the Action List and the Progress Bar. Towel [4], the user interface for the PExA time- and task-management tool [16], allows the user to delegate tasks to humans or software agents. RADAR automatically creates tasks from emails and fills in the forms used to do the tasks.

The SmartMail system automatically identifies sentences in emails that contain tasks and flags the emails in the inbox [5]. However, it does not classify the tasks by type nor does it add the tasks to a to-do list. RADAR implements both of those features as part of its task-centric user interface.

The Priorities system used email headers, content, and recipient availability to calculate the urgency of an email [14]. RADAR orders emails based upon the associated tasks, which include information from the email.

TASK-CENTRIC EMAIL PROCESSING

Initial Contextual Inquiries

We began exploring how to address the email overload problem by performing a set of contextual inquiries with approximately two dozen office workers to understand the problems that they encountered when handling their email. We quickly saw a pair of roles emerge, which we labeled *Initiator*, the person who needs assistance, and *Human Service Assistant* (HSA), the person who can provide the help. HSAs regularly received email requests that are performed by filling out forms, so their work serves as a good example for informing the design of the Action List interface. Observations of HSAs processing requests revealed that they would regularly save a small set of related tasks and then perform them together as a group. They claimed that this saved them time by sharing the overhead costs of connecting to different IT systems among multiple tasks.

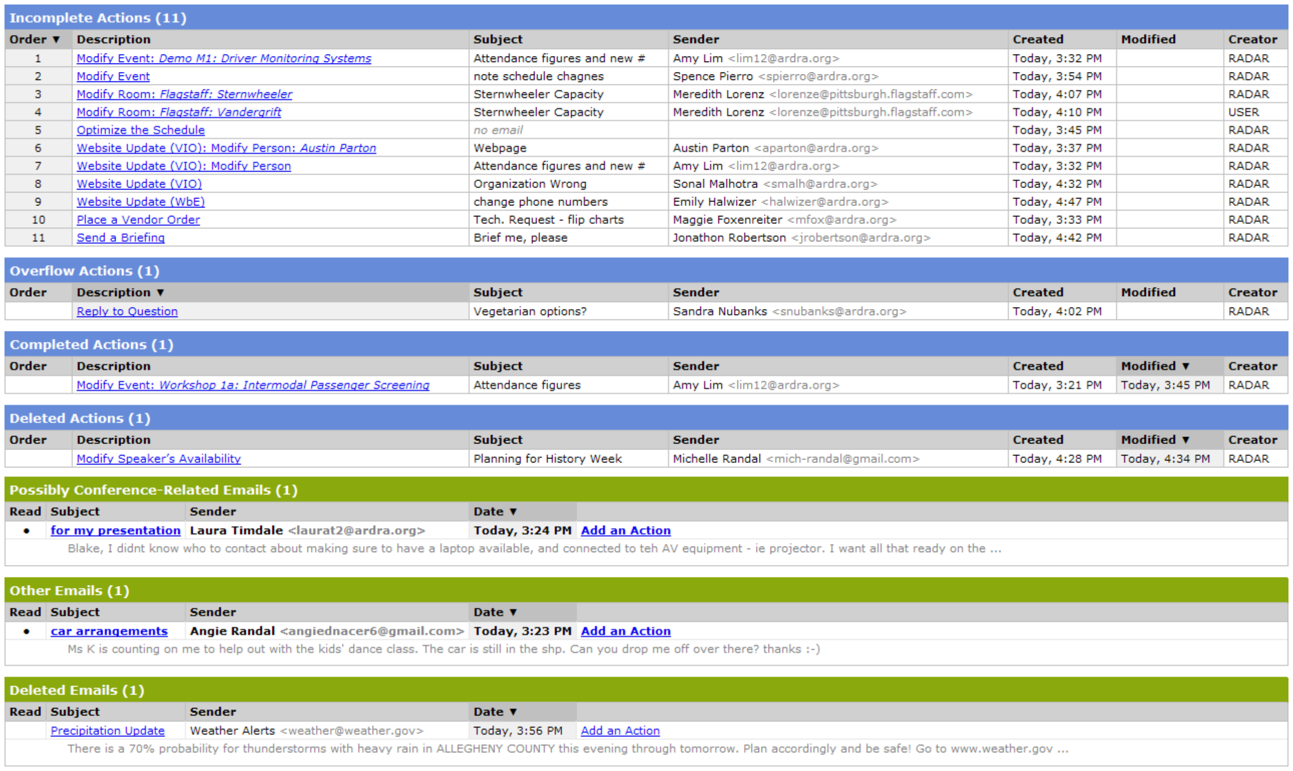
Inspired by these observations, we designed the Action List as a mechanism for transforming the processing of email requests from the traditional email-centric workflow to a task-centric workflow. This new perspective offered two distinct advantages over current email systems for an HSA. First, by providing a view of all tasks to be processed, HSAs could focus on completing one type of task at a time. We hypothesized that this would reduce the amount of context switching taking place, allowing HSAs to work more efficiently. Second, by interacting with a task-level view of incoming requests, HSAs could better prioritize their work.

We employed an iterative design process, building multiple working versions of the system. This approach helped us to assess both what kinds of errors the agents would make and to learn how well our users could recognize and repair these errors. One important discovery from the usability testing was that the term "action" works better than "task," so we used the former term in the user interface. However, we will continue to use the term "task" throughout this paper, except when referring to specific user interface elements.

The Action List

The Action List design supports a mixed-initiative interaction style for creating and completing tasks contained within emails. The Email Classifier examines the content of each email for evidence that it contains any requests of the eight known task types [25]. The Email Classifier dumps all available tokens and knowledge features of the email into one bag-of-words model and uses a regularized logistic regression algorithm, which scales to thousands of labels [26]. When it finds sufficient evidence for a given task type, it applies the label for that task type to the email. The classifier considers the evidence that supports each task type independently, and so it applies zero, one, or more different labels to each email. However, it cannot determine if an email contains multiple tasks of the same type. To improve classification performance, Scone [7], which is RADAR's knowledge base, provides additional ontological information that is not contained in the email's content. Examples include basic facts, such as "the Connan room is in the University Center," and higher-level concepts, such as "a peanut is kind of food that people might be allergic to."

The RADAR evaluation uses novice users, who may have difficulty effectively judging whether the email classifier's labels are correct. We were concerned that too many false positives might confuse them, so we tuned the classifier to favor precision over recall. An examination of the classifier's behavior showed that it did perform as desired. The latest evaluation had 123 emails, which contained 102 task labels. The classifier correctly found 47 task labels and incorrectly suggested 6 other task labels (false positives): *precision* = 0.887 (47/53) and *recall* = 0.461 (47/102).



(a) Incomplete Actions (11)

	Order ▼	Description	Subject	Sender	Created	Modified	Creator
(a.i)	1	Modify Event: *Demo M1: Driver Monitoring Systems*	Attendance figures and new #	Amy Lim <lim12@ardra.org>	Today, 3:32 PM		RADAR
	2	Modify Event	note schedule chagnes	Spence Pierro <spierro@ardra.org>	Today, 3:54 PM		RADAR
(a.ii)	3	Modify Room: *Flagstaff: Sternwheeler*	Sternwheeler Capacity	Meredith Lorenz <lorenze@pittsburgh.flagstaff.com>	Today, 4:07 PM		RADAR
(a.iii)	4	Modify Room: *Flagstaff: Vandergrift*	Sternwheeler Capacity	Meredith Lorenz <lorenze@pittsburgh.flagstaff.com>	Today, 4:10 PM		USER
	5	Optimize the Schedule	*no email*		Today, 3:45 PM		RADAR
	6	Website Update (VIO): Modify Person: *Austin Parton*	Webpage	Austin Parton <aparton@ardra.org>	Today, 3:37 PM		RADAR
(a.iv)	7	Website Update (VIO): Modify Person	Attendance figures and new #	Amy Lim <lim12@ardra.org>	Today, 3:32 PM		RADAR
	8	Website Update (VIO)	Organization Wrong	Sonal Malhotra <smalh@ardra.org>	Today, 4:32 PM		RADAR
	9	Website Update (WbE)	change phone numbers	Emily Halwizer <halwizer@ardra.org>	Today, 4:47 PM		RADAR
	10	Place a Vendor Order	Tech. Request - flip charts	Maggie Foxenreiter <mfox@ardra.org>	Today, 3:33 PM		RADAR
	11	Send a Briefing	Brief me, please	Jonathon Robertson <jrobertson@ardra.org>	Today, 4:42 PM		RADAR

(b) Overflow Actions (1)

Order	Description ▼	Subject	Sender	Created	Modified	Creator
	Reply to Question	Vegetarian options?	Sandra Nubanks <snubanks@ardra.org>	Today, 4:02 PM		RADAR

(c) Completed Actions (1)

Order	Description	Subject	Sender	Created	Modified ▼	Creator
	Modify Event: *Workshop 1a: Intermodal Passenger Screening*	Attendance figures	Amy Lim <lim12@ardra.org>	Today, 3:21 PM	Today, 3:45 PM	RADAR

(d) Deleted Actions (1)

Order	Description	Subject	Sender	Created	Modified ▼	Creator
	Modify Speaker's Availability	Planning for History Week	Michelle Randal <mich-randal@gmail.com>	Today, 4:28 PM	Today, 4:34 PM	RADAR

(e) Possibly Conference-Related Emails (1)

Read	Subject	Sender	Date ▼	
•	**for my presentation**	**Laura Timdale** <laurat2@ardra.org>	**Today, 3:24 PM**	**Add an Action**
	Blake, I didnt know who to contact about making sure to have a laptop available, and connected to teh AV equipment - ie projector. I want all that ready on the ...			

(f) Other Emails (1)

Read	Subject	Sender	Date ▼	
•	**car arrangements**	**Angie Randal** <angiednacer6@gmail.com>	**Today, 3:23 PM**	**Add an Action**
	Ms K is counting on me to help out with the kids' dance class. The car is still in the shp. Can you drop me off over there? thanks :-)			

(g) Deleted Emails (1)

Read	Subject	Sender	Date ▼	
	Precipitation Update	Weather Alerts <weather@weather.gov>	Today, 3:56 PM	Add an Action
	There is a 70% probability for thunderstorms with heavy rain in ALLEGHENY COUNTY this evening through tomorrow. Plan accordingly and be safe! Go to www.weather.gov ...			

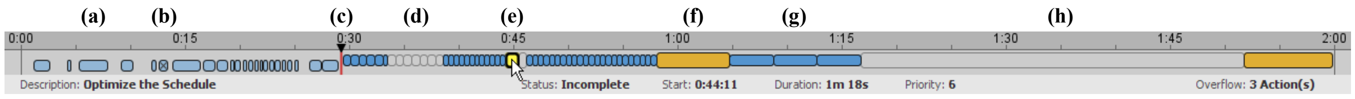
Figure 1: The RADAR Action List provides a task-centric view of an email inbox. The "Incomplete Actions" (a), "Overflow Actions" (b), and "Completed Actions" (c) tables list the tasks contained within email messages, allowing the user to sort by task-centric properties. The three email tables contain emails for which no tasks have been created (e, f, and g).

For each task label applied to an email, RADAR creates a task object, which is managed by the Task Manager [11]. Information stored includes the web form for that kind of task, if applicable. As we observed for real-life HSAs, in RADAR, many tasks require filling out web-based forms. RADAR's Natural Language Processor [17] attempts to specify task-specific parameters in the form, including the database record that the form should modify, if applicable. If RADAR can identify the record, then it will also try to fill in the other fields in the form. This novel integration of a to-do list with the forms for completing the tasks removes unnecessary steps from the process of performing a task.

The resulting tasks are displayed by the Action List, which provides a task-centric view of the user's email inbox (see Figure 1). The Action List allows a user to inspect the tasks that RADAR created, add ones that were missed, delete ones that should not have been created, and launch web pages to perform some of the tasks. The Action List contains seven tables divided into two groups: the first for tasks, and the second for emails (see Figure 1). The task group contains four tables that list "Incomplete" (a), "Overflow" (b), "Completed" (c), and "Deleted" (d) tasks. Tasks that the user has yet to perform are split between the Incomplete and Overflow table, with the latter table containing tasks that the MCA (to be described later) recommends that the user should skip due to time constraints. An email message will appear multiple times in the Action List if it is associated with multiple tasks of the same type (a.ii and a.iii), multiple tasks of different types (a.i and a.iv), or both. Tasks completed by the user appear in the Completed table, which provides the user with a record of their progress and allows them to go back and revisit previous tasks.

The tabular task display allows the user to sort their tasks with respect to task-centric properties such as "Description" or "Order" (by clicking on the column title), in addition to standard email-centric properties such as sender, subject, and date. The "Description" column sorts tasks alphabetically by type and within each type by the date of the associated email. When an email contains multiple tasks of different types, each of those tasks will be grouped with the other tasks of the same type when the table is sorted by the Description column. The other columns use a standard sort order based upon their data type.

The second set of tables (e, f, and g) display emails that are not associated with any non-deleted tasks. These tables are designed like a traditional inbox, with columns for the subject, sender, and date. The row for each email includes an excerpt from the beginning of the email body to help the user determine whether an email contains a task without opening the email. The user opens an email by clicking on either the subject or the "Add an Action" link. The email display includes the header and body sections along with the list of tasks that the user can add to the email.



Description: **Optimize the Schedule** Status: **Incomplete** Start: **0:44:11** Duration: **1m 18s** Priority: **6** Overflow: **3 Action(s)**

Figure 2: The Progress Bar shows completed (a) and deleted (b) tasks to the left of the current time (c), and the suggested schedule to the right. Noncritical tasks are blue (a, b, and g), critical tasks are orange (f), and expected tasks are gray (d and h). Details about the highlighted task (e) are shown in the status bar at the bottom.

The emails are divided among the three tables. The "Possibly Conference-Related Emails" table contains emails that RADAR thinks may contain tasks but for which it could not confidently identify the exact task type (e). This partial classification focuses the user's attention on emails likely to contain tasks without risking errors that might result if RADAR incorrectly classified the task as being of a particular type. The second table contains other emails that RADAR did not identify as task-related (f). The third table contains emails that the user deleted (g).

PROVIDING TASK ORDERING ADVICE

The evaluation of the RADAR 2.0 system showed that the task-centric workflow enabled by the AI technologies helps users. However, user performance varied significantly. Based upon analyses of the data logs, we hypothesized that some users had had difficulty finding a high-level strategy for completing the work. The novice users likely lacked meta-knowledge about tasks such as task importance, expected task duration, and task ordering dependencies. An expert user with that knowledge should be able to make good decisions about which tasks to work on at any given time and which tasks to skip when time is limited.

The MCA, a new component for RADAR 3.0, was built to address this problem by providing guidance about the order in which to work on a set of tasks. The MCA was designed to supports both near-term deadlines on the order of 1–8 hours, which would be encountered during a typical HSA's workday, and situations in which the amount of work exceeds the time allotted. The MCA learns task models upon which the advice is based by passively observing experts performing similar tasks. The MCA includes a novel visualization called the Progress Bar (see Figure 2), which shows a suggested schedule for performing incomplete tasks. The MCA's goal is to provide advice that improves performance and reduces overall performance variance, hopefully cutting off the long-tail of poor performance.

User Interfaces for Suggesting a Schedule

The primary advice provided by the MCA is a suggested schedule, which specifies an order in which to perform outstanding tasks. The MCA suggests which tasks to skip when it calculates that not enough time remains to perform all incomplete tasks. The MCA learns which tasks are generated after other tasks are completed, and so it also adds such "expected" tasks to the schedule. Including the expected tasks in the schedule provides the user with a more realistic understanding of upcoming work and eliminates major changes to the schedule that would otherwise occur when an expected task actually becomes necessary. Additionally, the MCA identifies "critical" tasks, which are particularly important for the user to complete. The suggested schedule is displayed by the Progress Bar (see Figure 2) and in the "Order" column of the Action List (see Figure 1).

The Progress Bar (see Figure 2) appears at the bottom of the screen just above the Windows Taskbar and always remains on top without obscuring other windows. Time is represented on the horizontal axis, which in this case spans two hours. An inverted black triangle and a vertical red line represent the current time (c), which increases from left to right. Each rectangular box represents a task. Task boxes to the left of the current time represent completed (a) or deleted (b) tasks, providing a record of the user's progress so far. The width of a task box represents the time that the user spent working on that task. The suggested schedule is visualized by the task boxes to the right of the current time. The width of each of these task boxes represents the amount of time that the MCA expects the task to require. Blue task boxes represent noncritical tasks (g). Orange task boxes, which are also slightly taller, represent critical tasks (f). Gray boxes represent expected tasks (d); expected critical tasks appear as taller gray boxes (h). The user can quickly inspect any task by moving the mouse over its task box (e), which updates the status bar at the bottom to show the task's description, status, actual/planned start time, actual/planned duration, and priority. That task, along with all other tasks of the same type, is drawn with a thicker border to allow the user to see where that type of task is distributed throughout the schedule. Double-clicking on a task box opens the corresponding task. The number of overflow tasks, which are the ones the MCA proposes to skip due to time constraints, appears at the bottom right.

MCA advice also appears in other parts of the RADAR user interface. First, the Action List's "Order" column shows the position of each future task within the suggested schedule (see Figure 1(a)). Only tasks in the "Incomplete Actions" table are included in the suggested schedule; the "Order" column is blank in the other tables. Sorting the "Incomplete Actions" table by the "Order" column shows the schedule as an ordered to-do list. Second, tasks that the MCA suggests that the user skip are shown in the "Overflow Actions" table. Third, after the user completes or deletes a task, RADAR redisplays the task's form in a finished state to provide feedback that the command succeeded. This confirmation screen also includes a link to the next suggested task in the schedule, which allows the user to navigate to the next task that the MCA recommends without having to return to the Action List. Finally, the MCA displays popup warning dialogs when the user significantly deviates from the suggested schedule. The warnings are issued if the user works on a critical task much earlier than experts did (Early Critical), if the user has not yet started working on a critical task by the time most experts had (Late Critical), or if the user starts working on a critical task that is not the next critical task on the suggested schedule (Wrong Critical).

Training the MCA

The MCA learned a model of expert behavior by passively observing experts performing tasks using the same user interfaces that test participants will later use. To train the system, experts did the two-hour study using a version of the system for which the MCA learning components were watching rather than recommending. Other AI components operated normally. For example, the Email Classifier had analyzed the emails to identify tasks. The training used three different email sets (none of which was the test email set), which provided variability to prevent overtraining.

Since there were no actual experts for the test tasks, we trained RADAR team members who had not worked on the MCA to be experts. People with detailed knowledge of how the MCA worked might behave in a way that aids the learning algorithm and hence would not be representative of real experts. We taught the experts how to use the RADAR components with the same instruction that novice test participants received. Additionally, the experts had significant exposure to the problem domain, detailed knowledge of some of the fixed stimuli (for example, available buildings), and were given more time to practice, all of which are consistent with the idea that they have more experience performing the test tasks. We also gave them some high-level strategy advice that a real expert would know based on knowledge of the scenario and evaluation methodology.

The primary goal of the MCA training process was to provide the MCA with an opportunity to infer the high-level strategy from passive observations of experts using that strategy to perform the two-hour study. The MCA learns the following information. First, the average *duration* to complete a task, along with the variance, is recorded per task type. Second, the MCA identifies *critical* tasks, which are tasks with a small number of instances and for which the mix of tasks changes after the critical task is completed, and computes an expected completion time for each critical task. Third, it learns task *phases*, which are the mix of tasks between critical tasks. Fourth, the MCA infers *prerequisites* among task types by looking for transitions between task types that are observed to occur with high probability. Each high probability transition is encoded as a directed edge within a partial ordering of the task types. Fifth, the MCA learns a *contextual task ordering*, which predicts the ordering among individual tasks. Sixth, the MCA computes *generate dependencies* among task types, which are the expected number of tasks of each type created after the completion of a task, with self-looping edges being allowed. Collectively, this information, which is computed using a variety of statistical machine learning algorithms, forms the MCA's learned model of expert behavior.

Generating the Suggested Schedule

Given the learned model and the current collection of tasks (both completed and incomplete) the MCA computes a suggested schedule for performing the tasks within the remaining time. Its goal is to produce a schedule representative of how an expert would perform the same collection of tasks. The process works as follows. First, the *generate dependencies* are used to create expected tasks. Second, the *prerequisites* (task-type ordering constraints) and output of the *contextual task ordering* predictor (individual-task ordering constraints) are passed into an Advice Integrator. Each constraint includes a weight that the Advice Integrator uses to compare against other constraints to produce a consensus schedule. The consensus schedule is a total ordering of the incomplete tasks, including expected tasks. Third, the Task Shedder [22] uses the learned task *durations* and task importance, along with the observed user's speed relative to the experts' speed, to decide which tasks to shed from the consensus schedule. The Task Shedder's algorithm generally sheds noncritical and expected tasks while shortening the planned duration of compressible tasks, but it never reorders individual tasks. The Task Shedder outputs the suggested schedule, which is presented to the user in the Progress Bar and Action List user interfaces.

EVALUATION

We evaluated RADAR using a conference planning test to determine how effective it is at assisting novice users based upon learned models of expert performance. This section describes the study design and then reports results from the evaluation of the RADAR 3.0 system.

The Conference Planning Test

A key challenge was designing an evaluation that measured how well RADAR reduced email overload. Since the project was planned to run for several years, we wanted to conduct carefully controlled, repeatable user studies that would allow us to measure progress over time. We initially considered designing an evaluation that used people's actual email, but decided against it. Such an evaluation would make comparisons difficult because of the vast differences among people's workloads. Additionally, any study would have to carefully protect the privacy of both the participants and those with whom they communicated via email.

Project members, in cooperation with external evaluators, developed a system-wide user test to evaluate how well RADAR's user interface and AI technologies assist a novice user. This section provides an overview of the test; a complete description can be found elsewhere [19]. The evaluation was designed to present participants with a challenging email overload workload that satisfied the following criteria. First, the tasks should be heterogeneous, just like in real-life. Second, some tasks could be handled quickly, while others would require significant effort. Third, dependencies should exist between some tasks, and doing those tasks out of order should result in wasted work or incorrect results. Fourth, email-centric properties such as subject lines should not be very helpful for grouping similar tasks or discerning efficient task orders. Fifth, users should not be expected to finish the test within the allotted time, so as to prevent ceiling effects in performance results.

The test presents participants with a simulated conference-planning scenario. Participants assume the role of the conference planner, filling in for the regular planner, Blake, who is indisposed. The simulated four-day, multitrack conference has keynotes, plenary talks, banquets, paper sessions, poster sessions, workshops, and so forth. Participants in our study must handle the outstanding conference planning tasks which have arrived in email, including many requests from the conference attendees. Blake's inbox contains these emails, which can be categorized as follows:

- **Scheduling:** Participants must update the database of event *constraints* (A/V requirements, meal preferences, attendee availabilities, and so forth) and conference room *properties* using an appropriate web form.
- **Website:** Attendees request changes to their contact information on the conference website. The participants must also update the website with the latest schedule.
- **Informational:** Attendees request information about the conference, generally concerning how the schedule has changed. The participants must author a reply email.
- **Vendors:** Attendees specify meal preferences and A/V requirements for events which then have to be forwarded to vendors using the vendor's web forms.
- **Briefing:** The conference chair requests a briefing summarizing the participant's progress at the end of the test. The inbox contains just one such request.

Participants also must deal with a conference crisis, which involves the loss of use of a significant number of rooms in which conference events had already been scheduled. Participants now need to find new rooms for the conference and adjust the schedule such that each event is placed in a room that satisfies the event's constraints, such as capacity, available equipment, seating arrangement, and so forth.

The tasks satisfy the criteria described earlier. Some tasks, such as informational requests, can be handled in around a minute, whereas other tasks, such as updating the conference schedule on the website, can take 15 minutes. Some emails contained over 20 tasks. An example of an intra-task dependency is that participants should handle the scheduling emails before updating the schedule on the website. Additionally, updating vendor orders requires submitting requests and waiting for email responses.

The test emails included anonymized real emails and fabricated ones, the latter necessary in part to make the emails consistent with the simulated world [19]. A team of undergraduate English majors was employed to create a detailed backstory email corpus, independent messages detailing one or more tasks, and noise messages, which were unrelated to the conference. The students were given a series of story arcs, guidelines, and a handful of characters with some specific assigned personalities (for example, formal, annoying, and so forth). Each study used the same backstory email corpus. However, to increase the validity of the tests, each study used a different *email set* of task and noise emails. An outside consultant customized those emails for each study. The consultant designed the emails to have comparable difficulty and task distribution. Between studies the stacks differed in the exact nature of the crisis—the specific rooms and times lost—and the details of the other email requests. These emails were kept secret until the test, so AI components could not train on them.

The test is designed to be hard—and it is. Over the past three years, approximately 400 people participated in the four major evaluations of different versions of RADAR; an additional 300 more people participated in interim evaluations. None of these people, including RADAR researchers, have been able to complete all of the tasks within the allotted time. We therefore think this evaluation approximates what a real person experiences, where it is often impossible to handle in one sitting all the emails that are pending.

The email set for the RADAR 3.0 study had 123 emails, of which 83 contained 153 tasks. The number of tasks is greater than the 102 task labels mentioned earlier, since some emails contained multiple tasks of the same type. The other 40 "noise" emails were unrelated to the conference.

Method

Conditions

The test used a between-subjects design with a single independent variable, *Assistance*, which had three levels: *Without Learning*, *Without MCA*, and *With MCA*.

In the *Without Learning* condition, most of RADAR's intelligent components were disabled. Specifically, the Action List initially had no email-based tasks since the Email Classifier was disabled, all the MCA advice was disabled, and the Progress Bar only showed the completed or deleted tasks. The main differences from the Action List in Figure 1 were that the "Order" column, the "Overflow Actions" table, and "Possibly Conference-Related Emails" table were not displayed, and the action tables were initially empty.

In the *Without MCA* condition, all of RADAR's AI components were enabled except for the MCA. The Action List contained the tasks that the Email Classifier found, and the "Possibly Conference-Related Actions" table contained emails. Again, the Progress Bar only showed the completed or deleted tasks. This condition was similar to RADAR 2.0. The main differences from the Action List in Figure 1 were that the "Order" column and the "Overflow Actions" table were not displayed.

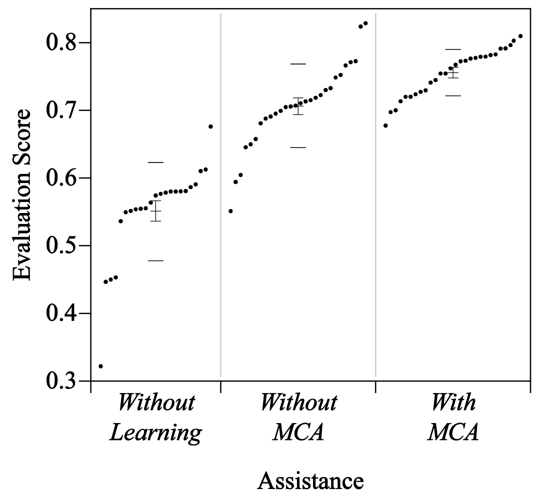


Figure 3: The evaluation scores show that the MCA advice in the *With MCA* condition significantly improved performance, reduced performance variation, and eliminated the long-tail of poor performance.

In the *With MCA* condition, all MCA functionality was enabled, as described in the previous sections.

Sessions

Each test session included up to 15 participants, who worked independently, and lasted up to 4.25 hours. In the first phase, which lasted 1.5 hours, participants learned about the test and received hands-on training with the software. After a break, participants started the two-hour testing session, which included another break after one hour. Then participants completed a survey and received payment, including extra payments for achieving specified milestones.

Participants

Participants were recruited from local universities and the general population using a human participant recruitment website. Participants were required to be between the ages of 18 and 65, be fluent in English, and not be affiliated with or working on RADAR. The study included 23 participants in the *Without Learning* condition, 28 participants in the *Without MCA* condition, and 28 participants in the *With MCA* condition. The number of participants varied among conditions since not all sessions yielded 15 usable data sets due to no-shows, participants who dropped out, participants who failed to make a good-faith effort, and software crashes or configuration issues that invalidated the data.

Results

Evaluation Score

An evaluation score, designed by external program evaluators, summarized overall performance into a single objective score ranging from 0.000 to 1.000, with higher scores reflecting better performance (for full details, see [10, 19]). It was important that this score be tied to objective conference planning performance rather than a technology-specific algorithm (for example, F1 for classification). This technology-agnostic approach allowed us to compare performance across conditions given any technology. The score was calculated from points earned as a result of satisfying certain conditions, coupled with costs and penalties. These included the quality of the conference schedule (for example, constraints met, special requests handled), adequate briefing to the conference chair, accurate adjustment of the web site (for example, contact information changes, updating the schedule on the website), costs for the rooms, food, and equipment for the conference, and penalties for requesting that others give up existing room reservations. The score coefficients were two-thirds for the schedule, one-sixth for website updating, and one-sixth for briefing quality. On this measure, *With MCA* participants clearly outperformed *Without MCA* participants, who in turn outperformed *Without Learning* participants (ANOVA, $F(2,76) = 83.7, p < 0.0001$) (also see Figure 3).

Assistance	**N**	**Mean**	**Std Dev**
Without Learning	23	0.550	0.072
Without MCA	28	0.706	0.063
With MCA	28	0.754	0.035

A subsequent Tukey post-hoc test found that the three conditions were significantly different from each other. All but 3 of the 28 *With MCA* participants earned higher scores than the average score of the *Without MCA* participants. Additionally, the standard deviation of the evaluation score dropped 44% from the *Without MCA* to *With MCA* condition and the long-tail of poor performance in the *Without MCA* condition disappeared, as we had hoped.

Email Classification and the Task-Centric Action List

We examined how well the task-centric user interfaces helped participants evaluate the suggestions of the Email Classifier. The following table lists the average number of tasks for each outcome.

	Without MCA		**With MCA**	
True Positives (TP)	47.0		47.0	
Viewed	43.6	100.0%	38.4	100.0%
Completed	38.4	88.0%	34.1	89.0%
Deleted	2.0	4.5%	1.9	4.8%
Ignored	3.3	7.5%	2.4	6.1%
Not Viewed	3.4		8.6	
False Positives (FP)	6.0		6.0	
Viewed	5.8	100.0%	5.3	100.0%
Completed	0.7	12.3%	0.8	14.3%
Deleted	2.7	46.9%	2.5	48.3%
Ignored	2.5	42.6%	2.0	37.4%
Not Viewed	0.6		0.7	
False Negatives (FN)				
Completed	4.5		2.0	
True Negatives (TN)				
Completed	4.3		1.8	
FP & TN Completed	**5.0**		**2.6**	

Of the 47 correctly classified tasks (TP) that participants inspected (Viewed), participants completed the majority of them, and rarely erroneously deleted any. Additionally, of the six incorrectly classified tasks (FP) that the participants inspected (Viewed), participants deleted or ignored the vast majority of them, only occasionally erroneously completing one. However, participants did not complete many tasks that the classifier missed (FN). They also created and completed some tasks when they should not have (TN: emails that actually did not contain any tasks). The *Without MCA* participants completed over twice as many TN compared with the *With MCA* participants (4.3 vs. 1.8; $t(54) = 2.5152$, $p < 0.02$). Overall, combining both commission errors (FP Completed + TN Completed), the *Without MCA* participants incorrectly completed on average 5.0 tasks, and the *With MCA* participants incorrectly completed 2.6 tasks.

In the *Without Learning* condition the Email Classifier was disabled, so participants had to inspect emails to find tasks. These participants correctly completed on average 43.7 tasks but incorrectly completed 10.3 tasks (equivalent to TN); the errors accounted for 19% of the completed tasks. While the *With MCA* participants made errors based upon incorrect AI suggestions, the participants without the assistance made up to four times *more* mistakes (10.3 vs. 2.6).

Effects of the MCA's Task Strategy Recommendations

Since participants earned significantly better evaluation scores in the *With MCA* condition than in the *Without MCA* condition, we examined the completed tasks to see how MCA advice may have impacted their scores.

The MCA identified five critical task types: "Optimize the Schedule" (run the Schedule Optimizer), "Publish the Schedule" (run script that updates the schedule on the conference website), "Bulk Website Update" (change the same type of information for many people on the website), "Reschedule Vendor Orders" (fix the vendors associated with events that moved in the schedule), and "Send a Briefing" (write a briefing for the conference chairperson). The following table shows the number of participants in each condition who completed each of the critical tasks at least once.

Task Type	Without MCA	With MCA
Optimize the Schedule	27	28
Publish the Schedule	27	28
Bulk Website Update	13	25
Reschedule Vendor Orders	3	6
Send a Briefing	25	28

The "Reschedule Vendor Orders" task takes about 30 minutes to complete so few participants in either condition finished it. However, the percentage of correctly scheduled vendor orders (a measure of partial progress on this task) was significantly higher in the *With MCA* condition than in the *Without MCA* condition (51% vs. 29%; $t(54) = 2.3400$, $p < 0.05$). Additionally, the percentage of money wasted on incorrectly scheduled vendor orders (another measure of partial progress) significantly dropped in the *With MCA* condition (30% vs. 66%; $t(54) = 3.3061$, $p < 0.01$).

Participants in the *Without MCA* condition completed more total tasks (65.5 vs. 55.3; $t(54) = 2.4770$, $p < 0.02$) and more noncritical tasks (54.0 vs. 44.9; $t(54) = 2.671$, $p < 0.02$) than the *With MCA* participants did. Yet, the *With MCA* participants earned higher scores, because they did the more important tasks rather than just doing more tasks.

The following table shows that participants generally complied with the critical task warnings that the MCA issued.

Task Type	Issued	Complied	%
Late Critical	93	83	89%
Wrong Critical	25	14	56%
Early Critical	1	0	0%
Total	112	97	83%

Compliance with the "Late Critical" warnings was high. However, participants did not allows follow the "Wrong Critical" alerts. Five of these participants seemed to be averse to quitting what they were currently working on. This could be exacerbated by the fact that participants are instructed that critical tasks are special, and therefore they might believe that finishing the current one is more important than following the warning's advice.

In the *With MCA* condition, the average position of a task in the suggested schedule at the time that it was finished (either completed or deleted) was 5.0. Finished tasks were in the top position 21% of the time and within the top five 62% of the time. Since the *Without MCA* condition does not provide a suggested schedule, we computed the position of the task in the Action List when it was finished. In the *Without MCA* condition, the average position of tasks when it was finished was 11.6. Finished tasks were in the top position 18% of the time and within the top five 37% of the time. Finally, we found no significant difference for the number of times that participants followed the "next suggested task" link (19.2 in *With MCA* vs. 17.8 in *Without MCA*; $t(54) = 0.3246$, n.s.).

Discussion

The participants clearly found the AI systems helpful in performing their tasks. They were able to understand the AI component's suggestions and override them when in error. We looked for reasons why participants did not seem to be following the MCA's recommendation for the specific next task to do. It appears that users often were skipping the top one or two tasks over and over, suggesting that they did not want to do those specific tasks for some reason. Thus, participants were relying on the MCA to give them strategic advice of an overall order, but felt comfortable looking within the top few recommendations. This lends support to our mixed-initiative user interface rather than one that just presented the next task. Our pop-up alerts for critical tasks also were successful in focusing the user's attention on critical tasks they were ignoring in the other views.

CONCLUSION AND FUTURE WORK

Now that the RADAR techniques have proven so successful in our lab study that simulated a HSA's workload, we are eager to transition these techniques to a real email system with online learning. The main hurdle will be making the AI components robust enough for use with real-world tasks and emails, and integrating the AI technologies and the user interface with the real forms that are used to perform the tasks. Additionally, the HSA workload represents a subset of the work performed by office workers, and we are interested to see how our techniques apply to other workloads.

ACKNOWLEDGMENTS

The authors thank Michael Freed, Geoff Gordan, Jordan Hayes, Javier Hernandez, Matt Lahut, Pablo-Alejandro Quinones, Nicholas Sherman, Stephen Smith, Fernando de la Torre, Pradeep Varakantham, Jigar Vora, Yiming Yang, Shinjae Yoo, and Gabriel Zenarosa. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

REFERENCES

1. Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D.G. and Ducheneaut, N. What a To-Do: Studies of Task Management Towards the Design of a Personal Task List Manager. *Proc. CHI*, ACM Press (2004), 735–742.
2. Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R.E. Quality Versus Quantity: E-Mail-Centric Task Management and Its Relation With Overload. *Human-Computer Interaction 20*, 1/2 (2005), 89–138.
3. Bellotti, V. and Thornton, J.D. Managing Activities with TV-ACTA: TaskVista and Activity-Centered Task Assistant. *Proc. SIGIR Workshop on PIM*, (2006), 8–11.
4. Conley, K. and Carpenter, J. Towel: Towards an Intelligent To-Do List. *Proc. AAAI Spring Symposium on Interaction Challenges for Artificial Assistants*, AAAI Press (2007), 26–32.
5. Corston-Oliver, S., Ringger, E., Gamon, M. and Campbell, R. Task-focused Summarization of Email. *Proc. ACL Workshop on "Text Summarization Branches Out"*, Association for Computational Linguistics (2004), 43–50.
6. Ducheneaut, N. and Bellotti, V. E-mail as Habitat: An Exploration of Embedded Personal Information Management. *interactions 8*, 5 (2001), 30–38.
7. Fahlman, S.E. Marker-Passing Inference in the Scone Knowledge-Base System. *Proc. KSEM*, Springer (2006), 114–126.
8. Faulring, A., Mohnkern, K., Steinfeld, A. and Myers, B.A. The Design and Evaluation of User Interfaces for the RADAR Learning Personal Assistant. *AI Magazine 30,* 4 (2009), 74–84.
9. Fink, E., Bardak, U., Rothrock, B. and Carbonell, J.G. Scheduling with Uncertain Resources: Collaboration with the User. *Proc. IEEE SMC*, IEEE Press (2006), 11–17.
10. Freed, M., Carbonell, J., Gordon, G., Hayes, J., Myers, B., Siewiorek, D., Smith, S., Steinfeld, A. and Tomasic, A. RADAR: A Personal Assistant that Learns to Reduce Email Overload. *Proc. AAAI-08*, AAAI Press (2008), 1287–1293.
11. Garlan, D. and Schmerl, B. The RADAR Architecture for Personal Cognitive Assistance. *IJSEKE 17*, 2 (2007), 171–190.
12. Gwizdka, J. TaskView: Design and Evaluation of a Task-based Email Interface. *Proc. CASCON*, IBM Press (2002), 4.
13. Horvitz, E. Principles of Mixed-Initiative User Interfaces. *Proc. CHI*, ACM Press (1999), 159–166.
14. Horvitz, E., Jacobs, A. and Hovel, D. Attention-Sensitive Alerting. *Proc. UAI 1999*, Morgan Kaufman (1999), 305–313.
15. Kumar, M., Das, D. and Rudnicky, A.I. Summarizing Non-textual Events with a 'Briefing' Focus. *Proc. RIAO*, Centre De Hautes Etudes Internationales D'Informatique Documentaire (2007).
16. Myers, K., Berry, P., Blythe, J., Conley, K., Gervasio, M., McGuinness, D., Morley, D., Pfeffer, A., Pollack, M. and Tambe, M. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine 28,* 2 (2007), 47–61.
17. Nyberg, E., Riebling, E., Wang, R.C. and Frederking, R. Integrating a Natural Language Message Pre-Processor with UIMA. *Proc. LREC Workshop on "Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP"*, (2008), 28–31.
18. Shneiderman, B. and Maes, P. Direct Manipulation vs. Interface Agents. *interactions 4,* 6 (1997), 42–61.
19. Steinfeld, A., Bennett, S.R., Cunningham, K., Lahut, M., Quinones, P.-A., Wexler, D., Siewiorek, D., Hayes, J., Cohen, P., Fitzgerald, J., Hansson, O., Pool, M. and Drummond, M. Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop. *Proc. PerMIS*, NIST (2007).
20. Steinfeld, A., Quinones, P.-A., Zimmerman, J., Bennett, S.R. and Siewiorek, D. Survey Measures for Evaluation of Cognitive Assistants. *Proc. PerMIS*, NIST (2007).
21. Stylos, J., Myers, B.A. and Faulring, A. Citrine: Providing Intelligent Copy and Paste. *Proc. UIST*, ACM Press (2004), 185–188.
22. Varakantham, P. and Smith, S. Linear Relaxation Techniques for Task Management in Uncertain Settings. *Proc. ICAPS*, AAAI Press (2008), 363–371.
23. Whittaker, S., Bellotti, V. and Gwizdka, J. Email in Personal Information Management. *CACM 49,* 1 (2006), 68–73.
24. Whittaker, S. and Sidner, C. Email Overload: Exploring Personal Information Management of Email *Proc. CHI*, ACM Press (1996), 276–283.
25. Yang, Y., Yoo, S., Zhang, J. and Kisiel, B. Robustness of Adaptive Filtering Methods in a Cross-Benchmark Evaluation. *Proc. SIGIR*, ACM Press (2005), 98–105.
26. Yang, Y., Zhang, J. and Kisiel, B. A Scalability Analysis of Classifiers in Text Categorization. *Proc. SIGIR*, ACM Press (2003), 96–103.
27. Zimmerman, J., Tomasic, A., Simmons, I., Hargraves, I., Mohnkern, K., Cornwell, J. and McGuire, R.M. VIO: A Mixed-initiative Approach to Learning and Automating Procedural Update Tasks. *Proc. CHI*, ACM Press (2007), 1445–1454.