RADAR: A Personal Assistant that Learns to Reduce Email Overload

Michael Freed[1], Jaime Carbonell[2], Geoff Gordon[2], Jordan Hayes[3], Brad Myers[2], Daniel Siewiorek[2], Stephen Smith[2], Aaron Steinfeld[2] and Anthony Tomasic[2]

[1]SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

[2]School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[3]Bitway Inc.
1700 Shattuck Ave., Suite. 320
Berkeley, CA 94709

Abstract

Email client software is widely used for personal task management, a purpose for which it was not designed and is poorly suited. Past attempts to remedy the problem have focused on adding task management features to the client UI. RADAR uses an alternative approach modeled on a trusted human assistant who reads mail, identifies task-relevant message content, and helps manage and execute tasks. This paper describes the integration of diverse AI technologies and presents results from human evaluation studies comparing RADAR user performance to unaided COTS tool users and users partnered with a human assistant. As machine learning plays a central role in many system components, we also compare versions of RADAR with and without learning. Our tests show a clear advantage for learning-enabled RADAR over all other test conditions.

Email and Task Management

Once widely hailed as the "killer app" for networked computing, email now gets more attention as a source of inefficiency, error, and stress. Complaints about excessive time spent processing, storing, finding, and collating messages are commonplace among office workers at every level and across organization type (Balter 1998). Business self-help books (e.g., Song et al. 2007) coach readers on regaining control of their lives from the "tyranny of email." Studies of *email overload* show that the problem can negatively impact work performance (Dabbish and Kraut 2006) and that it imposes substantial costs on organizations and national economies (Spira and Goldes 2007). Early exploration of the problem focused on the ever-increasing volume of email that users send and receive. More recent investigations emphasize the centrality of email in workplace tasks of all kinds, the resulting adoption of email for personal task management (Mackay 1988; Whittaker and Sidner 1996), and the many inadequacies of email client software for this purpose (Belotti et al. 2003).

Various projects have explored ways to reduce email overload by improving client software. Most focus on streamlining some aspect of message handling, e.g., by automatically filing messages into user-defined folders (e.g., Mock 2001), helping users quickly scan and decide what to do with unhandled messages (Cadiz et al. 2001), and identifying dependencies between messages in a conversational thread (Rohall et al. 2001). Belotti et al. (2003, 2005) have gone further and reconceptualized the email client in keeping with their view that "dealing with email and dealing with tasks and projects are indistinguishable." Their system, Taskmaster, combines email client and to-do list functionality with the aim of alleviating problems that their studies show typical users have with task management in email.

These efforts differ in approach, but share the assumption that a single application can serve as both a good email client and a good task management tool. This assumption seems suspect since users benefit at different times from both message-centric and task-centric interfaces, and hybrid designs typically entail compromise. Comments from Taskmaster prototype users (Belotti et al. 2005) suggest just such a tradeoff – e.g., complaints that screen real estate devoted to listing and detailing tasks left too little available to view messages. The limitations of a hybrid approach become more apparent when we consider integrating email not only with task management tools, but also with numerous application programs useful for carrying out email-derived tasks. Prospects for making one tool do it all seem remote.

An alternative approach follows the model of a trusted personal assistant who reads and processes messages on the boss's behalf, helping the boss to sustain focus on critical tasks and stay organized. The assistant uses whatever tools are appropriate – an email client to read, organize, and respond to messages; task or project management software to maintain a to-do list; diverse application programs to prepare or execute tasks. The assistant does not rely on engineered integration of these capabilities into a single tool but instead uses knowledge of user tasks and how the tools support those tasks to integrate their functions dynamically.

The RADAR project has developed a software-based personal assistant intended to help users cope with email overload as effectively as a human assistant. The system analyzes message text received by the user to distill out task-relevant information including new tasks elicited by a message. RADAR adds newly identified tasks to a displayed Task List and provides machine-learning-based guidance to the user on which task to do next. When the user indicates the intention to begin a task by clicking on it, RADAR invokes the appropriate application program and

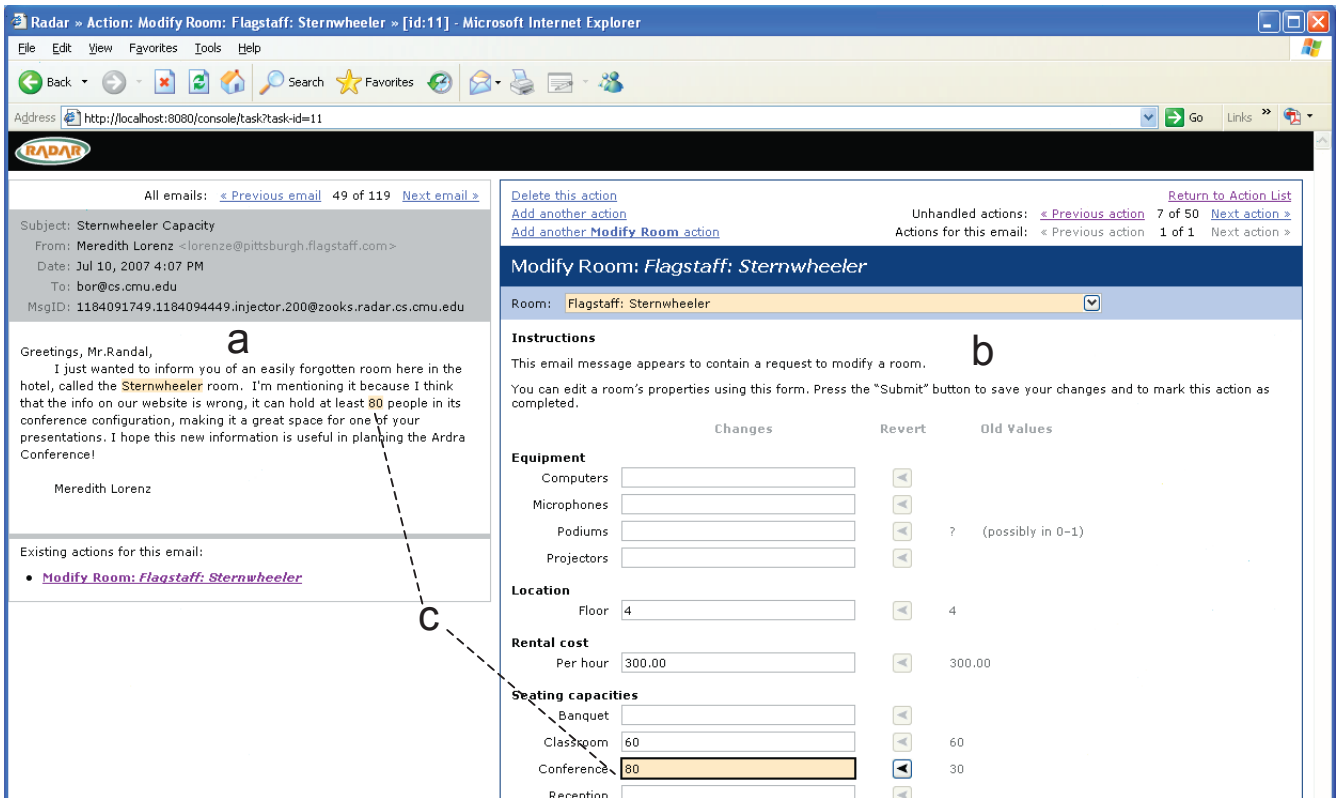

Figure 1. After the user selects a task to modify incorrect meeting room data from the Task List (not shown), Radar displays the email message (a) that originated this task alongside a database record modification form (b) used to carry out tasks of this type. Radar fills in form fields based on extracted email content (c), leaving it to the user to either accept or correct field values.

interacts with the user to assist execution (Figure 1). This paper describes how RADAR components are integrated to provide intelligent assistance and overviews results from system evaluation experiments designed to measure progress toward human-level assistance.

System Overview

The RADAR system consists of three sets of components (Figure 2). Message-Task Linking (METAL) components analyze email messages to identify new tasks, distill out other task relevant content and establish user-navigable links between task representations and message text. Using METAL output, Multi-task Coordination Assistance (MCA) components provide task-management support to the user such as maintenance of a to-do list and guidance on which item on the list to do next. Powertools (PT) are application programs similar to the tools people normally use to produce work products, but are enabled by METAL and knowledge from past interactions to provide task-aware user assistance.

Identifying Task-relevant Message Content

RADAR Message Task Linking (METAL) components find associations between email text and user task representations. Especially important are mechanisms that detect new tasks elicited by arriving email and extract specifying task parameters from message text. For example, a message announcing that corporate visitors will arrive at a particular date and time might imply a task to reserve a conference room starting just after their scheduled arrival. Task-relevance of a given message may vary for different recipients depending on their goals and responsibilities (e.g., an administrative staff person responsible for allocating meeting space might view the above email differently from someone merely interested in attending the meeting), so METAL mechanisms must learn what constitutes a task separately for each user.

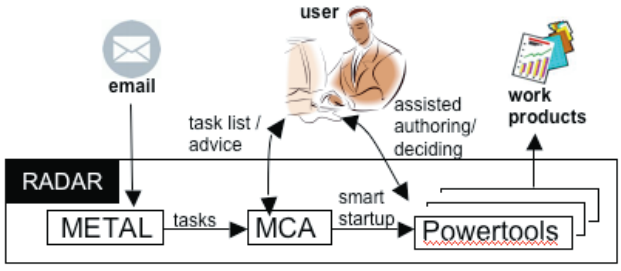

Figure 2. Radar System

The high-level process for detecting and specifying email-derived tasks is, in order: preprocess, annotate, classify, extract. Preprocessing includes stemming (normalizing text terms with the same morphological root) and removal of stop words, as is typical for reducing dimensionality of text for classification. Numerous natural language processing components then annotate the text. Some of these are domain independent. For example, RADAR's Time Expression Annotator identifies temporal expressions (e.g., "next week") in text and resolves (anchors) them to points or ranges in absolute time (Han et al. 2006). Other annotators are tied to particular users or task types. For example, RADAR includes an annotator that recognizes room names to support meeting-room scheduling tasks. In some cases, RADAR can learn to improve these annotators by observing as users perform tasks with appropriately instrumented application programs (Tomasic et al. 2007).

RADAR assumes that the types of typical user tasks are known and treats email task detection as a text classification problem. We used a regularized logistic regression suite of classifiers (based on body, headers, links) and combined their results (Yang et al. 2005). The classifiers were designed to be incrementally adaptive to accommodate the impact of real-world changes (e.g., new people, changing projects).

We obtained fairly good results on task-based classification (Figure 3) with *accuracy* (proportion of correct positive and negative classifications of an email message by task type) ≥ .90 for all task types and *F1*[1] *macro average* (a stronger measure generally considered more meaningful than accuracy) = .67. The question of whether this performance is good enough is best answered in the end-to-end system evaluation (see section below) where RADAR users, overriding METAL errors as needed, significantly outperformed users of off-the-shelf (COTS) office software.

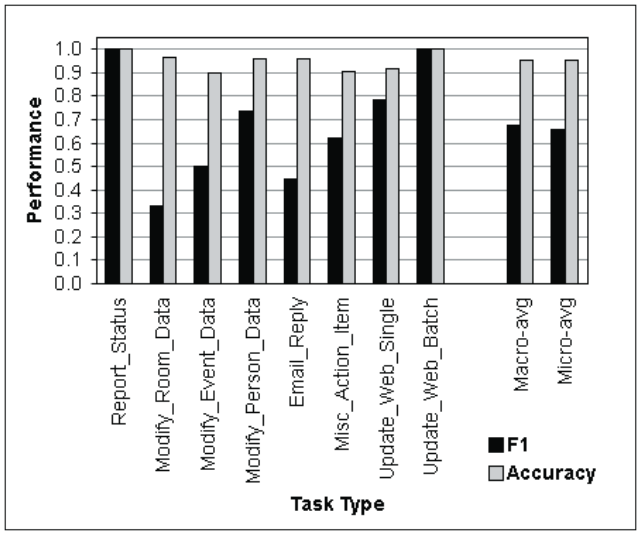

Figure 3. Task classification results

[1] F1 is the harmonic mean of *recall* (fraction of all existing tasks of a given type found by the classifier) and *precision* (fraction of correctly identified tasks of a given type in tasks returned by the classifier).

Once a message has been categorized by task type, RADAR extracts task-specific parameters from the annotated text. For example, given a message found to contain a task of type `schedule-meeting-room`, extractors would look for temporal expressions corresponding to the meeting start and end times. For a task type with known parameters, extraction can be considered a form-filling task (Cohen et al. 2005). RADAR's extraction method uses lightweight parsing as input to the trainable MinorThird system (Cohen 2004; Carvalho and Cohen 2006) that matches text terms to task fields.

Task Management Assistance

RADAR takes a layered approach to user-system integration and support for task management. At its core, the key integrative structure is an explicit representation of user/system tasks. The representation provides a basic accounting of all currently known tasks including their constraints (deadlines, dependencies with other tasks), associations to other media (email) and execution state. Through linkage to learned models of various task types, these representations provide a basis for anticipating resource requirements (e.g. necessary data inputs) and for inferring task execution profiles (expected durations, likely follow-on tasks). The set of currently known tasks is managed by the RADAR Task Manager (TM) and made accessible to the user through the RADAR Console. The TM provides basic services for defining/instantiating new tasks, monitoring task interactions and preconditions, managing execution status, and collecting relevant user action data and task results.

The current Task List is exposed to the user through the Console along with information and advice intended to help the user stay organized and focused. The Console also provides controls for selecting tasks and invoking tasks. Email-derived tasks are linked back to their originating messages. Users can easily inspect and edit task data to compensate for METAL detection errors, thereby facilitating assisted task execution provided by RADAR Powertools.

Layered on top of this integration infrastructure is the Multi-task Coordination Assistant (MCA). MCA tackles the broader problem of managing a busy user's workload. It uses learned knowledge about the types of tasks a user performs, accumulated user experience in performing these tasks, and learned preferences of specific users to provide several complementary forms of task management advice.

Attention management – One form of advice MCA provides is to draw user attention to tasks that have been

neglected or overlooked. The Attention Manager (AM) learns productive task sequences and workflows, tracks real-time progress, and alerts the user after detecting failure to progress at important, urgent tasks. Because interruptive reminders can themselves reduce productivity (Iqbal and Horvitz 2007) and can annoy users, the AM attempts to issue reminders at task boundaries (after completing one task, before starting the next) when user tolerance for interruption is highest (Smailagic et al. 2007).

Task Prioritization – A second form of advice provided by MCA addresses the basic question of what task to do next. The Task Prioritizer (TP) uses a Hidden Naïve Bayes classifier (Zhang et al. 2005) to learn a set of dynamic task ordering preferences in a given workflow setting from analysis of the actions of expert users. A simple voting method is then used to prioritize pending tasks facing the current user at any point. These priorities are conveyed as suggestions to the user within the Console's todo list and are updated periodically to reflect variable urgency.

Task Shedding – A third form of advice provided by MCA considers the question of where and how to cut corners when all tasks cannot be done within given time constraints. The Task Shedding (TS) Advisor (Varakantham and Smith 2008) uses learned models of domain structure (characterizing task types and workflow constraints) together with task prioritization information provided by TP to construct and maintain an overall task schedule. The schedule is represented using a Simple Temporal Problem with Preferences and Uncertainty model (Rossi et al. 2006). When expected workload is projected to exceed available time, a relaxed solution is computed that identifies which tasks to drop and which others to compress to minimize expected quality loss. The TS suggests this strategy to the user.

Task Execution Assistance

In office environments, carrying out a task typically involves using application software to create or modify work products in the form of, e.g., documents, spreadsheets, messages, and databases. These programs incorporate assistive automation such as spelling checkers and macro recorders, but are not task-aware and thus cannot provide assistance that depends on knowing the type, status, or interactions among user tasks. In RADAR, output from METAL components enables task-awareness – at least for recognized email-derived tasks; learning capabilities within an application make it possible to translate that awareness into task execution assistance. The RADAR system incorporates METAL-powered applications ("Powertools") focused on execution assistance for three kinds of tasks: information retrieval and update, resource scheduling, authoring status briefings.

Assisting Information System Update Tasks. Human assistants are often asked to look up or update data in an information system. For instance, human resource assistants process numerous requests to update personnel records; executive assistants process requests to schedule conference rooms or make travel arrangements. Processing a request typically involves reading it and then locating, completing, and submitting a form corresponding to the expressed intent. For example, upon reading a request to "Please change the phone number on John Doe's web page to 555-1212", a web site administrator would (1) select the right form to update records in the database back end to the web site; (2) select "John Doe" in the name field to identify the specific record to be modified, possibly using a pull-down menu; (3) enter a modified value into the phone number field; (4) press the "submit" button to execute the specified modification.

The RADAR Virtual Information Officer (VIO) component attempts to automate this process. In the best case, METAL mechanisms accurately identify the task type (`Change Person`) and task parameters (`name: John Doe; phone: 555-1212`) in the request message; VIO then retrieves and fills out the form on the user's behalf and presents the results to the user for verification. If correct, the user only needs to confirm VIO's top form choice (ranked highest on a displayed list) and press "Submit" to complete the task. If not, the user modifies the incorrect values before submitting the form.

The first three processing steps in the phone number modification example above exemplify key challenges addressed by VIO machine learning algorithms (including METAL capabilities specific to VIO). For step 1, a k-way boosted decision tree algorithm is used to generate a ranked list of possible forms. In experimental evaluation on a webmaster corpus (Zimmerman et al. 2007), this approach performed very well (mean reciprocal rank > .90) and better than tested alternatives. For step 2, entity instance selection, VIO uses a Naïve Bayes classifier (mean reciprocal rank > .85). For step 3, form field completion, VIO uses a conditional random field model trained to generate an extraction model. Performance varies greatly by form field ($0.0 < F1 < 1.0$), depending most strongly on the number of training labels. Each algorithm is trained by a process of "domestication" wherein labels on source message text are generated when the user repairs incorrect VIO form-type, entity or form-field predictions (Tomasic et al. 2007). Note that in VIO, high precision is favored over high recall, because the cost to the user of approving a correct prediction is much lower than the cost of repairing an incorrect prediction, and the cost of handling a missed prediction is exactly the cost that the user would incur without assistance.

Assisting Resource Scheduling Tasks. A common task for human assistants is to manage limited organizational resources such as conference rooms and meeting equipment. Resource management can become especially demanding when there are multiple resources to manage,

multiple interacting resource requests, and overall demand is high. The RADAR Space-Time Planner (STP) and CMRadar components provide automated assistance in such cases. STP provides scheduling optimization capabilities, incorporating a new optimization algorithm (Fink et al. 2006) designed for the kinds of mixed-initiative allocation problems and uncertainty emphasized in the RADAR evaluation (see section below). Optimizers are only as good as the information supplied to them. The RADAR project has placed particular emphasis on the problem of timely, incremental capture of relevant scheduling information from different sources and by different means including

- Automatic recognition of schedule-relevant data and constraint changes in e-mail traffic
- Learning regularities in the task environment, such as the likelihood of success in seeking new resources that change the scheduling problem to one that is less constrained (Oh and Smith 2004)
- Improved representation of uncertainty [Bardak et al. 2006a] to more accurately capture data and constraints
- Active learning of critical data and constraints to reduce uncertainty and thus enable improved optimization (Bardak et al. 2006b)

Assisted Authoring of Status Reports. Assistants are often responsible for tracking progress and reporting status of some larger task such as completing a project or preparing for a meeting. This can be especially demanding and error-prone in high workload situations when the effort to record and report takes valuable time away from the effort to accomplish primary tasks. The RADAR Briefing Assistant (BA) assists users in authoring status reports that summarize what has and has not yet been accomplished. The creation of high-quality reports requires extensive knowledge capture, inference and synthesis capabilities including (i) instrumentation of the activities performed by the user, (ii) knowledge of relevant activities for reports, including activities that should have occurred, (iii) a learned model of the language and the level of abstraction best used in the summary (e.g., "most updates were done" versus "7 of 9 updates were done"), and (iv) a learned model of the relative importance of various parts of a status report. System instrumentation in all RADAR components tracks activity data and logs it in the TM that is used by the BA. When the user begins a task to generate a status briefing (represented on the RADAR-maintained to-do list), the system summarizes past activity into a list of "bullet points" ranked and abstracted in accordance with the BA's learning model (Kumar et al. 2006).

System Evaluation

As a 5-year project under DARPA's Personalized Assistant that Learns (PAL) program, RADAR is evaluated yearly to measure machine learning research progress with particular focus on "learning in the wild" – i.e., learning from passive observation or natural interaction with the user. Project members set two additional evaluation objectives: (1) measure the utility of RADAR compared to state-of-practice alternatives, thus providing a basis for deciding when the technology is mature enough for practical use and (2) obtain guidance on how best to direct component-level research to advance this goal. Here we briefly overview experimental design and results from tests run in two consecutive years (year 2 and year 3 of the project). See Steinfeld et al. (2007a) for a more detailed discussion.

Experiment Design

The experiments included four conditions varying the tools available to human subjects. In the first condition, subjects used a version of RADAR that had undergone a period of simulated use, allowing learning mechanisms to train the system and (presumably) improve performance. We refer to this condition as *RADAR + Learning* (abbreviated "+L"). In the "-L" condition, subjects used an untrained version representing RADAR prior to any learning opportunity. "COTS" subjects used commercial off-the-shelf tools instead of RADAR to perform the task. Each "COTS+A" participant was randomly paired with one of 10 human assistants, each a confederate member of the experiment team who shared the workload and helped the subjects manage tasks (e.g., by prompting them to do urgent tasks). The experiments tested four hypotheses:

+L > COTS: subjects assisted by RADAR would beat those using COTS tools

+L > COTS+A: subjects with RADAR would beat those with a human assistant

+L > -L: RADAR would provide better assistance with learning enabled than with learning disabled

ΔL increasing: performance attributable to learning would increase in each yearly test (from system improvements, not additional training)

The subjects' overall task was to take over organizing a fictional academic conference, substituting for a now-incapacitated organizer ("Blake Randal") who had made significant progress but left some things unfinished or incomplete. Subjects were instructed to go through Blake's email inbox (119 messages) to learn what else needed doing and come as close as possible to completing remaining tasks. Some messages had clear associated tasks (e.g., a request to change a misspelled name on the conference web site) or implied tasks (e.g., information that a speaker would be arriving late in the conference, implying a need to reschedule the session and then post the revised schedule to the website), while others contained irrelevant "noise" messages. In addition, an unexpected "crisis" described in the message stream (e.g., that a block of rooms with scheduled sessions is no longer available)

partially invalidated existing plans, requiring significant replanning.

Materials developed to support the test included the email corpus and a simulated world, the latter specifying information about the conference (e.g., sessions to be scheduled), the physical environment (e.g., rooms where sessions might be placed), and people (e.g., speaker names and contact info). The conference itself was a 4-day, multi-track meeting with more than 130 talks and posters, complete with exhibit hall, social events, poster sessions, tutorials, workshops, plenary talks, and a keynote address. Speakers and other people defined in the scenario were assigned email addresses, phone numbers and, in many cases, fax numbers, website addresses, and organizations.

Subjects used a web browser to access information about the physical space (campus center, academic building, off-site hotel) presented in university web pages accessible from the subject's home page. Other static web content included Blake's original schedule, a conference planning manual, and manuals for tools used by the subjects. Subjects were also given access to a realistic web-based, "university approved" vendor portal where goods and services such as audio-visual equipment, security, and catering could be ordered for the conference. Email receipts including hyperlinks to modification/cancellation pages and computed prices were delivered to the subject's mail client in real time. Developing this experimental framework required substantial time and resources. Effort to make it available for use by other research projects is under way (Steinfeld 2008).

Overall score in the range [0.000 - 1.000] was a weighted sum of sub-scores on three work products: the conference schedule (based on constraints met, special requests handled, correct vendor orders made, and so on); conference web site (accuracy and completeness); and status briefing to the conference chair. Score criteria, weights (2/3 for the schedule, 1/6 each for the web site and briefing), and possible score penalties (e.g., for overspending budget) were determined by outside evaluators. In addition, outside evaluators designed key elements of each test including the email message stacks, ensuring that RADAR could not be designed around particular test elements.

Recruited subjects had no relevant domain experience or knowledge of RADAR, but were required to be computer literate and fluent in English. Each was run through approximately 4 hours of testing – 2 for subject training and 2 for time on task. During the training period, subjects were informed of the scenario, their responsibilities, task scoring criteria, and payout schedule (designed to align subject priorities with scoring criteria). They were trained to use tools appropriate for their assigned test condition and guided through practice examples of all core toolset features. For subjects in the +L and –L conditions, these tools included RADAR components described earlier in this paper: STP and CMRadar for creating the conference schedule; VIO for web site updates and vendor orders; BA for authoring status updates.

Results

As shown by the table of results below, the first three hypotheses were easily met with p-values below 0.0001. Thus, subjects assisted by RADAR+L (with learning) outperformed subjects using COTS tools, with and without an assistant, and subjects using RADAR-L (without learning). The difference in performance across years between non-learning versions of RADAR was not statistically significant. However, performance in the learning condition (as measured by the delta between the learning and non-learning condition) improved significantly (t-ratio 2.9, $p<0.0034$) from year 2 to year 3 with the contribution of learning to overall performance increasing by 41%.

Condition	Year 2		Year 3	
	N	Final_Score	N	Final_Score
COTS	18	0.402	31	0.409
COTS+A	n/a	n/a	20	0.525
Radar –L	18	0.509	42	0.534
Radar +L	18	0.630	32	0.705

More generally, our evaluation showed that task-aware application programs and task management tools can improve user performance and that automated detection of tasks and task-relevant information can be particularly helpful in reducing the impact of email information overload. In addition, post-test surveys (Steinfeld et al. 2007b) found that RADAR users had a far more favorable test experience than subjects using comparable COTS tools. In particular, RADAR users were more confident that they had done tasks well, found tasks easier to complete, felt more immersed in the test, and assessed themselves (correctly) as having completed more work.

Conclusion

RADAR technology is intended for use in real-world environments. Is it ready? Results of our evaluation studies are encouraging about its potential to speed tasks and improve user experience under high email load. Positive survey responses indicate that user acceptance does not depend on AI providing near-perfect task awareness and that good user interface and interaction design can compensate. Perhaps the biggest unknown is whether users will tolerate relatively low performance early on as the system learns to detect task-relevant message content and provide meaningful assistance. Continued research on RADAR focuses on ways to reduce the training interval (e.g., Donmez and Carbonell, 2008), provide more intelligent assistance at an early stage of training, and dynamically restructure user interaction to reflect current system competence.

Several efforts to incorporate RADAR capabilities into non-research tools are under way. METAL technology has been incorporated into the trouble report processing for the U.S. Navy's F/A-18 Automated Maintenance Environment. Tests on more than 6500 such reports (2 months worth) showed good performance classifying the kind of trouble-handling task (F1 micro-average = .84) and determining priority on a 1-5 scale (F1 micro-average = .72). This level of performance was seen as likely to improve report-handling efficiency and led to rapid integration of the technology into operational Navy systems. Other efforts focused on integrating RADAR into work systems and general productivity tools are ongoing.

Acknowledgments

The authors thank the more than 100 researchers and developers who have contributed to RADAR. Special thanks to Andrew Faulring and William Cohen for comments and suggestions on this paper. This material is based upon work supported by the Defense Advanced Projects Research Agency (DARPA) under Contract No. FA8750-07-D-0185.

References

Balter, O. 1998. Electronic Mail in Working Context. PhD Thesis. Royal Institute of Technology, Stockholm, Sweden.

Bardak, U., Fink, E., Carbonell, J. 2006a. Scheduling with Uncertain Resources Part 2: Representation and Utility Function. *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pages 1486-1492.

Bardak, U., Fink, E., Martens, C., Carbonell, J. 2006b. Scheduling with Uncertain Resources Part 3: Elicitation of Additional Data. *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pages 1493-1498.

Belotti, V., Ducheneaut, M., Howard, M., Smith, I. 2003. Taking Email to Task: The Design and Evaluation of a Task Management Centered E-Mail Tool. *Proc. 2003 Conference on Computer/Human Interaction*, ACM Press, 345-352.

Belotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R. 2005. Quality vs. Quantity: Email-centric task management and its relations with overload. *Human-Computer Interaction,* 20, 2/3, 89-138.

Cadiz, J. J., Dabbish, L., Gupta, A., and Venolia, G. D. 2001. *Supporting Email Workflow*. Technical Report MSR-TR-2001-88. Redmond, WA: Microsoft Research.

Carvalho, V., Cohen, W. 2006. Improving Email Speech Act Analysis via n-gram Selection. *HLT/NAACL ACTS Workshop*.

Cohen, W. 2004. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net

Cohen, W., Minkov, E. and Tomasic, A. 2005. Learning to Understand Web Site Update Requests. *Proc. 1995 International Joint Conference on Artificial Intelligence.*

Dabbish, L., Kraut, R. 2006. Email Overload at Work: An Analysis of Factors Associated with Email Strain. *Proc. 2006 Conference on Computer Supported Collaborative Work*, ACM.

Donmez, P., Carbonell, J. 2008. Paired Sampling in Density-Sensitive Active Learning. *Proc. 10th International Symposium on Artificial Intelligence and Mathematic*s, Florida.

Fink, E., Jennings, M., Bardak, U., Oh, J,. Smith,, S. Carbonell, J. 2006. Scheduling with Uncertain Resources: Search for a Near-Optimal Solution. *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pages 137-144.

Han, B., Gates, D., Levin, L. 2006. From Language to Time: A Temporal Expression Anchorer. *Proc. 13th International Symposium on Temporal Representation and Reasoning (TIME 2006)*.

Iqbal, S.T., Horvitz, E. 2007. Disruption and Recovery of Computing Tasks: Field Study, Analysis and Directions. *Proc. 2007 Conference on Computer/Human Interaction.*

Kumar, M., Garera, N., Rudnicky, A. 2006. A Briefing Tool that Learns Report-writing Behavior", *IEEE International Conference on Tools with Artificial Intelligence (ICTAI),* Washington D.C., USA.

Mackay, W.E. 1988. More than Just a Communication System: diversity in the use of electronic mail. *Proc. 1988 Conference on Computer Supported Collaboartive Work*. ACM.

Mock, K. 2001. An Experimental Framework for Email Categorization and Management. *Proc. SIGIR 2001*. New York: ACM.

Oh, J., Smith, S. 2004. Learning User Preferences in Distributed Calendar Scheduling. *Proc. 5th International Conference on the Practice and Theory of Automated Timetabling,* Pittsburgh PA.

Rohall, S.L., Gruen, D., Moody, P., Kellerman, S. 2001. Email Visualizations to Aid Communications. *Proc. 2001 Conference on Information Visualization*, IEEE. 12-15.

Rossi, F., Venable, K. B., Yorke-Smith, N. 2006. Uncertainty in Soft Temporal Constraint Problems: A general framework and controllability algorithms for the fuzzy case. *JAIR* 27:617–674.

Smailagic, A., Siewiorek, D., De la Torre, F., Pradhan, P., Shah, A., Vigar, J. 2007. *Task Interruptibility Analysis.* Technical Report, Institute for Complex Engineered Systems, Carnegie Mellon University, Pittsburgh, PA.

Song, M., Halsey, V., Burress, T. 2007. *The Hamster Revolution: How to Manage Your Email Before It Manages You*, Berrett-Koehler.

Spira, J., Goldes, D. 2007. Information Overload: We Have Met the Enemy and He is Us. Report by Basex Inc.

Steinfeld, A. 2008. Airspace web site, http://www.cs.cmu.edu/~airspace

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., Drummond, M. 2007a. Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop. *Proc. NIST Performance Metrics for Intelligent Systems Workshop*.

Steinfeld, A., Quinones, P.-A., Zimmerman, J., Bennett, S. R., Siewiorek, D. 2007b. Survey Measures for Evaluation of Cognitive Assistants, *Proc. NIST Performance Metrics for Intelligent Systems Workshop.*

Tomasic, A., Simmons, I., Zimmerman, J. 2007. Learning Information Intent via Observation. *Proc. International World Wide Web Conference (WWW)*.

Varakantham, P., Smith, S.F., "Advising Busy Users on How to Cut Corners", Technical Report CMU-RI-08-17, The Robotics Institute, Carnegie Mellon University, April 2008.

Whittaker, S., Sidner, C. 1996. Email overload: Exploring personal information management of email. *Proc. 1996 Conference on Computer/Human Interaction*. ACM.

Yang, Y., Yoo, S., Zhang, Z., Kisiel, B. 2005. Robustness of Adaptive Filtering Methods in a Cross-benchmark Evaluation. *Proc. 28th Annual International ACM SIGIR* Conference (SIGIR 2005), Brazil.

Zhang, H., Jiang, L., Su, J. 2005. Hidden Naive Bayes. *Proc. AAAI*, 919–924. AAAI Press.

Zimmerman, J., Tomasic, A., Simmons, I., Hargraves, I., Mohnkern, K., Cornwell, J., McGuire, R. 2007. VIO: A mixed-initiative approach to learning and automating procedural update tasks. *Proc. 2007 Conference on Computer/Human Interaction.*