Do Humans Adapt Their Emails So That Agents Can Understand?

Pragnesh Jay Modi, Kerry Hannan, Manuela Veloso
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
{pmodi, khannan, veloso}@cs.cmu.edu

Abstract

Natural language processing is a difficult problem and is currently a technical barrier in building personal assistant agents that aim to interact with humans in natural language. One possible solution is to rely on humans to restrict or adapt their language into more computer friendly ways. We study the feasibility of this approach in the context of a personal assistant agent that parses emails relevant to meeting scheduling in order to assist the user in calendar management tasks. We design an experiment in which humans are given email writing tasks and then given feedback on how well those emails were understood by an agent. We wish to see if the humans learn to adapt their email writing to what is understandable by their agent.

Introduction

Personal agents are software assistants that aim to reduce human cognitive load by automatically classifing, processing and replying to incoming and outgoing emails on behalf of their user (Maes 1994; Mitchell *et al.* 1994; Chalupsky *et al.* 2001). Motivated by the high volume of email related to meeting scheduling, the CMRadar project (Modi *et al.* 2004) is concerned with building agents to assist with calendar management tasks in an office environment. Our vision is that a CMRadar agent assists its user by processing incoming and outgoing scheduling-related emails written in natural language, negotiating and soliciting calendar information from other users, and when appropriate, making autonomous scheduling decisions.

As a first step of the process, a CMRadar agent must parse an email written in natural language to create a machine-readable logical representation in preparation for further downstream processing. Natural language provides a great deal of freedom in expressing information, arguably so much so that constructing an agent to extract key pieces of information from emails written in natural language is a very difficult problem (Jurafsky & Martin 2000). This difficulty creates a significant hurdle to building personal agents to assist in meeting scheduling in an office environment.

Rather than attempting to build an agent that can cope with a wide range of natural language input, in this paper we investigate the opposite question of whether and to what extent we can rely on the human to adapt to the limited capabilities of the agent. Indeed, the repetitive nature of scheduling a meeting via email suggests that users could perhaps learn to write meeting scheduling emails in a more computer-friendly i.e. less ambiguous, format for an agent.

Zoltan-Ford (Zoltan-Ford 1991) investigates this question in the context of users who make queries to a database in natural language. The author hypothesizes that techniques such as *modeling* and *shaping* can be used to make human computer users reduce the variability of their natural language queries and hence increase the accuracy of the database query processor in understanding them. Modeling refers to the human tendency to mimic the language of those they are communicating with, while shaping refers to the human tendency to adapt their language based on past success and failure at communicating (Zoltan-Ford 1984).

Following the work of Zoltan-Ford, we study whether human users who know that an agent will be reading their emails learn to adapt the way they write based on feedback (shaping). Many other successful systems have tried to capitalize on this sort of user-adaptability to put a smaller burden on the part of the agent. These systems include information retrieval from a database and IBMs scribble writing for PDAs. Although one could provide the user with a manual that describes acceptable input, it is well-known that there are significant Human-Computer Interaction advantages to letting users figure out these rules themselves.

We conduct an experimental study aimed at investigating the feasibility of the above approach. We designed an experiment in which humans were given the task of writing meeting scheduling emails and then provided with feedback on how well the agent understood their emails. Specifically, after writing an email users are presented with a list of information the agent extracted from their email, which they must then edit if anything is incorrect. The idea was that this process will clue in the participant to the agents capabilities. In this way, the user receives feedback about what the agent can or cannot understand. Users are told in advance that their goal is to write their emails in a way so as to minimize the number of changes they have to make to the list of information extracted by the agent, but are not told explicitly how to do so.

The results of our study are mixed. On the positive side, we found that extraction errors made by the agent were reduced over time as the human test subject continued to interact with the agent. On the negative side, there was significant variance in the results. Our number of human test subjects was limited which prevented obtaining statistically significant data. In the remainder of this paper, we describe our experimental design and the results obtained.

The CMRadar Agent

CMRadar is developed as a personalized agent that interacts with other users or agents. It assumes that multiagent interaction in calendar meeting scheduling occurs through email message exchange. The Extractor component of CMRadar is responsible for parsing email messages into a logical representation for further processing. This logical representation, which we call a *Template*, holds the key pieces of task-relevant information contained in an email for the meeting request or reply to a request. We first describe the Template format and then the Extractor component.

Template Data Format

For communication with other agents or humans, templates are converted to and from natural language emails using an Extractor and Generator. An example Template is shown in Figure 1. This Template representation is used internally within the CMRadar agent for more complex downstream reasoning about meetings. Fields in the template included (but were not limited to):

- **Start Time**: Time the meeting starts.
- **Start Date**: Date the meeting starts.
- **End Time**: Time the meeting ends.
- **End Date**: Date the meeting ends.
- **Purpose**: Description of the meetings purpose.
- **Duration**: Duration of the meeting.
- **Attendants**: A list of people who will attend the meeting.

Extractor

The extractor used during this experiment was a pattern-based parser written in Perl. A partial specification of this parser is shown in Figure 2. The extractor searched through a natural language email and its meta-data and attempted to fill in the fields of a Template.

As a simplification to real life meeting scheduling, the extractor only handled one possible start time/date and end time/date. The extractor was developed and updated based upon 2 practice trials of the experiment (described later) to handle popular patterns that were not already accounted for and were considered reasonable. The majority of these updated patterns included variants of temporal expressions such as noon 2pm, and 3:00PM. The goal of the extractor was to be reasonable but not perfect. Temporal expressions that were rejected include three thirty, and 13h40.

```
(template
  (meeting-id MT5) (msg-id MGS1205)
  (timestamp 2003-12-17[15:04 -0500])
  (initiator sfs@cs.cmu.edu)
  (duration 3600) (location NSH1305)
  (time-slots
    (time-slot
      (earliest-start-time 2003-12-17[15:00 -0500])
      (latest-finish-time 2003-12-17[16:00 -0500])
      (start-time 2003-12-17[15:00 -0500])
      (finish-time 2003-12-17[16:00 -0500])
      (priority 1)
      (status confirmed)))
  (attendants
    (attendant (id sfs@cs.cmu.edu) (level 1.0))
    (attendant (id mmv@cs.cmu.edu) (level 1.0)))
  (purposes
    (purpose (predefined-kind project-meeting)
             (description “Radar project meeting”)
             (special-note nil))))
```

Figure 1: Example of a CMRadar template

```
duration:  [ “for” | “last” ] <digits> [ “hr(s)” | “min(s)” ]
timeslots: [ “at” | “before” | “after” ] <time-exp>
           : <time-exp> “to” <time-exp>
           : <time-exp> “-” <time-exp> //ex: ”10:00 - 11:00”
time-exp:  <1-2 digits>”:”<2 digits> //ex: ”10:00”, ”6:00”
           : <1-2 digits>”:”<2 digits> <tag>
           : <1-2 digits> <tag> // ex: ”10 am”, ”1 pm”
tag:       [ “am” | “a.m.” | “pm” | “p.m.” ]
```

Figure 2: Example grammar rules for converting emails to Templates.

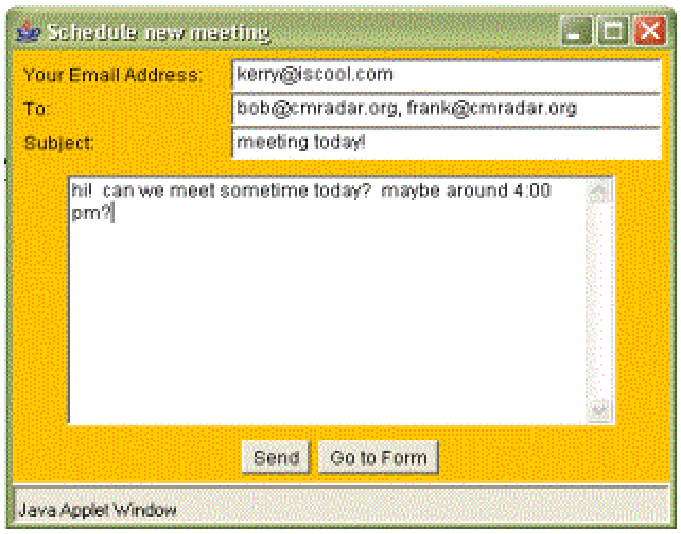
Experiment Design

Test Subjects

Our results come from 19 subjects where the data was collected over a period of 2 weeks. Subjects were solicited through online bboards within Carnegie Mellon University. Most participants were students, though the participant pool included staff and faculty as well.

Procedure

To initiate the experiment, participants were given the URL of the experiment website and allowed to read the instructions and to continue onto the tasks at their leisure. There was no time limit and no restrictions on completing the tasks in one single sitting. However, participants were randomly given 1 of 10 tasks, one after the other, so taking time in between tasks required keeping an open browser window. The experiment itself was written as a Java applet, so participants could access the experiment from any computer supporting a standard version of java and any java-enabled web browser such as Internet Explorer or Mozilla. The experiment would have a total of 4 windows for each task: instructions for the experiment (window 1), a description of the task (window


Schedule new meeting

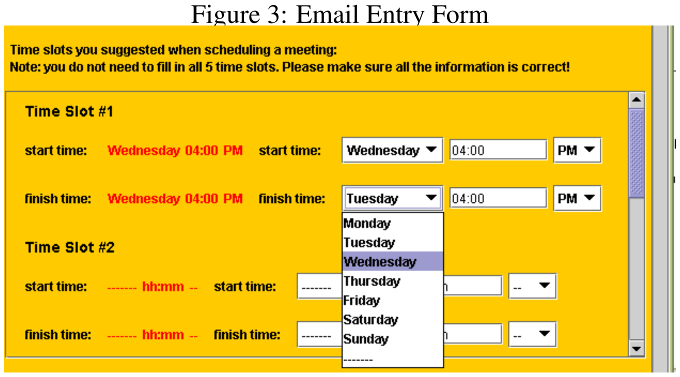
Your Email Address: kerry@iscool.com

To: bob@cmradar.org, frank@cmradar.org

Subject: meeting today!

hi! can we meet sometime today? maybe around 4:00 pm?

Figure 3: Email Entry Form


Time slots you suggested when scheduling a meeting:
Note: you do not need to fill in all 5 time slots. Please make sure all the information is correct!

Time Slot #1

start time: Wednesday 04:00 PM start time: Wednesday 04:00 PM

finish time: Wednesday 04:00 PM finish time: Tuesday 04:00 PM

Time Slot #2

start time: ------- hh:mm -- start time: ------- --

finish time: ------- hh:mm -- finish time: ------- --

Figure 4: Extraction Correction Form

2), a simulated email window (window 3), and finally an editable form for corrections (window 4). Participants were always allowed to refer back to the experiment instructions, the task description, and their original email when correcting the extraction template.

Task Descriptions

Subject participants were given 10 sequential email writing tasks. These tasks formulated the parameters of a scenario in an office environment in which the user would be required to write an email to other attendees in order to schedule a meeting. The 10 tasks were presented in a randomized order to each participant. Below is an example task given to a test subject:

Example Task: *Next Thursday, an important visitor (stu@cmradar.org) is coming, and we'd like for you and your colleague (chloe@cmradar.org) to meet with him. You are too busy to meet in the morning on Thursday, so you will have to try to schedule this meeting at another time.*

Special care was taken to ensure that the task descriptions did not force or lead the participant to write his or her email in a particular syntax. First, participants were not permitted to copy and paste the text from the task description. Secondly, no temporal expressions were explicitly mentioned in any of the task description prose. If the task description were to have included temporal expressions such as 3:00 PM, then the participant might be less inclined to use other temporal expressions such as 3pm. Thus, no specific times were mixed in the prose of the task descriptions.

After reading the task, the user was shown an email window in which to write her email in natural language form (see Figure 3). The user was instructed that this email was intended to be read by the other meeting participants and also to be read by her agent.

Feedback to Subjects

After completing the email writing, participants were given two types of feedback: a structured, editable template describing the information that was extracted from the email and a score. We describe both forms of feedback below.

After each email writing task was completed, the user was presented with the extraction template obtained by running the Extractor engine on the email. As shown in Figure 4, the extraction template had 2 columns: the left column contained a list of all the fields the extractor was investigating, while the right column contained the same items, but allowed the participant to edit these fields to make corrections. In this way, the user could see what the parser extracted and their own correction side-by-side. The structure and syntax of the displayed feedback were also carefully constructed to avoid confusing the participant. For example, the duration of the proposed meeting was internally stored in seconds. However, the parser would not accept the unlikely description of a meeting lasting for 3600 seconds. Instead, users were shown only expressions the parser could understand (though the parser certainly understood more than it displayed in the feedback to the user).

Participants were also given a score during the experiment, which counted how many modifications to the extraction template a participant made. After correcting an extraction template, the user would be shown how many modifications he/she had just made, and how many modifications have been made total. Along with this counter, an emoticon and color codings were added to enforce an excellent, mediocre, or bad score. Below is are two examples of the score given to a user.

```
Ex:  You made 0 changes :-)
Ex:  You made 2 changes :-(
```

Results

Our results count the number of changes that were made to the extracted template. A change constitutes any kind of edit a user makes to the extracted information. In the entire extraction template, there were 8 changes that could be made, detailed as follows: Start-Time, Start-Time AM/PM, Start Date, End-Time, End-Time AM/PM, End Date, duration, and meeting attendants. Additional change categories were added to account for the fact that a user may change only a portion of these fields. The AM/PM field is especially susceptible to being wrong when the rest of the Start time or End Time field is not since a user may not include AM or PM in the original email.

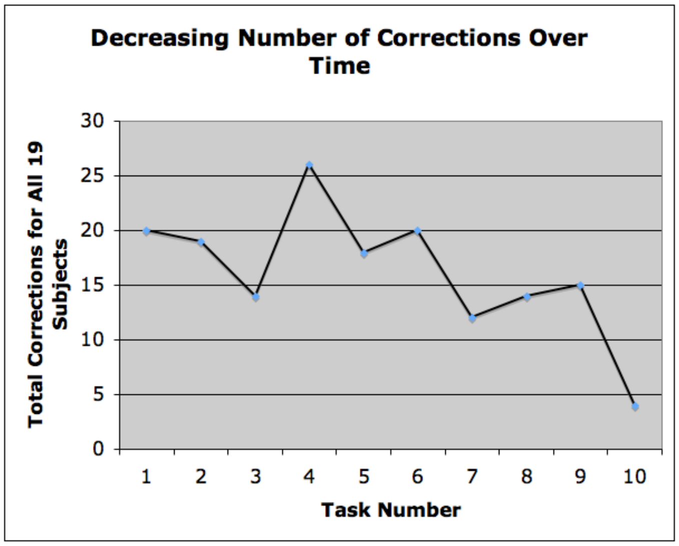


Figure 5: Number of corrections made to all fields

Implicit in our experiment design was the assumption that initially the humans would write their emails in a way in which the Extractor agent would perform very poorly and the user would need to make many changes. To our surprise, we found that the number of errors made by the Extractor were generally low, even initially. This was problematic for determining if the number of changes would decrease during the course of the experiment. However despite this, we see in Figure 5, the number of changes that users made decreased over time. Halfway through the task experiment, participants were 20% less likely to make a change to the template than during the first half of the experiment.

We also investigate the changes users made based on particular fields. Figures 6, 7 and 8 show the number of corrections made to the End-Time, Duration and AM/PM fields respectively. We see a similar pattern of decreasing corrections over time. This is especially clear with the AM/PM field.

In all graphs, we see a spike in the number of changes in the middle of the experiment, somewhere between Task 3 and 6. We hypothesize that this pattern is caused by test subjects initially writing emails in a conservative way, then "testing the boundries" of the agent by writing more creatively. When the subject sees that the agent fails to extract successfully, the subject goes back to what worked. Hence, we see the spikes in the number of corrections.

On the negative side, we see that these results are not statistically significant since the overall number of changes made was rather low and the number of test subjects was also low. Also, it is informative to view the results of a post-experiment survey in which participants were asked a) how satisfied they were with the performance of their agent at understanding their emails, and b) if they felt they had changed the way they wrote their emails during the course of their experiment. On a scale of 0 10 (0 being not satisfied at all and 10 being completely satisfied), the average score was 7.2 and 13 out of the 19 participants reported that they did not change the way they wrote their email very much to

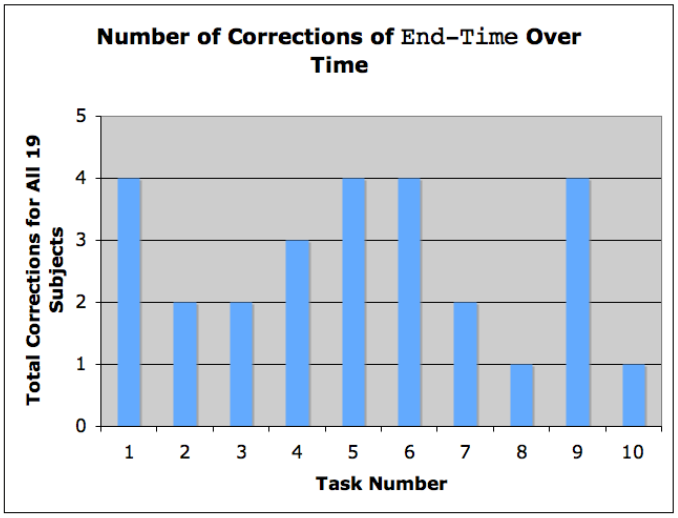


Figure 6: Number of corrections made to End-time field

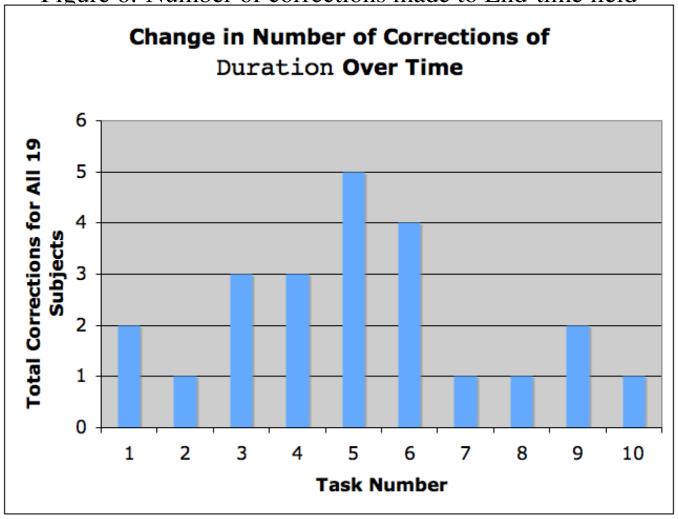


Figure 7: Number of corrections made to duration field

accommodate for the email. We conclude that since users were mostly satisfied with the performance of their agent, it makes sense that many users would not have felt they changed their writing style at all.

Conclusions

The conclusion to be drawn from this study is that limited natural language capabilities may not be as significant a problem in building personal assistant agents as perhaps initially thought. We saw that a simple parser agent performed surprisingly well on emails that were written by humans. We suspect the cause of this was that the humans knew in advance that the emails they write will be parsed by an agent and so they wrote them in a simpler way than they otherwise would. To confirm this hypothesis, a control case experiment is needed in which emails written by humans who do not know that an agent will be reading them are given to the parser and its accuracy is compared to the results presented here.

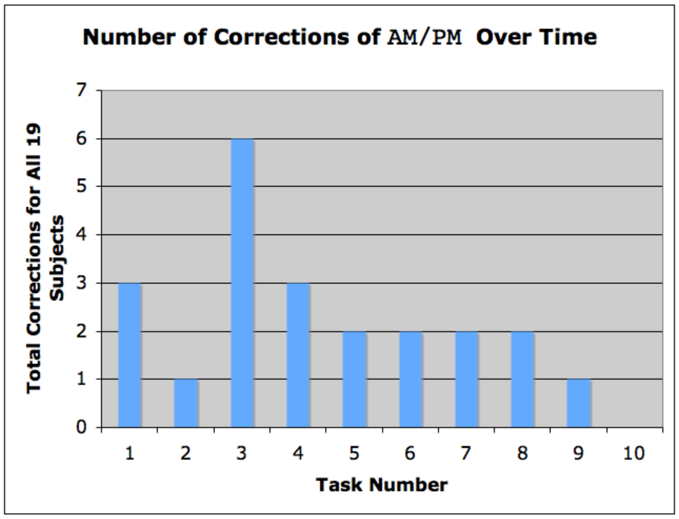


Figure 8: Number of corrections made to AM/PM field

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA or the Department of Interior-National Business Center (DOI-NBC).

References

Chalupsky, H.; Gil, Y.; Knoblock, C.; Lerman, K.; Oh, J.; Pynadath, D.; Russ, T.; and Tambe, M. 2001. Electric elves: Applying agent technology to support human organizations. In *Proceedings of Innovative Applications of Artificial Intelligence Conference*.

Jurafsky, D., and Martin, J. H. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics.* Prentice-Hall.

Maes, P. 1994. Agents that reduce work and information overload. *Communications of the ACM* 37(7).

Mitchell, T. M.; Caruana, R.; Freitag, D.; ott, J. M.; and Zabowski, D. 1994. Experience with a learning personal assistant. *Communications of the ACM* 37(7):80–91.

Modi, P. J.; Veloso, M.; Smith, S.; and Oh, J. 2004. Cmradar: A personal assistant agent for calendar management. In *Agent Oriented Information Systems, (AOIS) 2004*.

Zoltan-Ford, E. 1984. *Language shaping and modeling in natural-language interactions with computers*. Ph.D. Dissertation, The Johns Hopkins University.

Zoltan-Ford, E. 1991. How to get people to say and type what computeres can understand. In *Int. Jour. Man-Machine Studies*, volume 34.