

Towards Intelligent Assistance for To-Do Lists

Yolanda Gil and Varun Ratnakar

USC Information Sciences Institute

4676 Admiralty Way, Marina del Rey, CA 90292

{gil, varunr}@isi.edu

ABSTRACT

Assisting users with to-do lists presents new challenges for intelligent user interfaces. This paper presents a detailed analysis of to-do list entries jotted by users of a system that automates tasks for users that we would like to extend to assist users with their to-do entries. We also present four distinct stages of interpretation of to-do entries that can be accomplished and evaluated separately. A system that has good performance in any of these four stages can provide intelligent assistance that is useful to users.

Author Keywords

User interfaces, to-do lists, automated assistance, natural language interpretation, knowledge acquisition, knowledge collection from web volunteers, office assistants.

ACM Classification Keywords

H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

To-do lists are ubiquitous, whether in our handheld organizers or in traditional form as paper lists and sticky notes. From a user perspective, to-do lists can be viewed as “entry points” that suggest to the user to engage in the tasks therein, often carrying information about deadlines and importance level [9]. Studies have shown that to-do lists are the most popular personal information management tools (used more than calendars, contact lists, etc.), used by more than 60% of people consulted about their usage of personal information management [8]. To-do lists are external artifacts that augment human cognition in that they serve as memory enhancers by reminding people of what needs to be done. Norman [12] points out that such external artifacts often transform the tasks that users do into new sets of tasks. He uses to-do lists to illustrate this, pointing out that although they are indeed helpful as

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IUI'08, January 13-16, 2008, Maspalomas, Gran Canaria, Spain.

Copyright 2008 ACM 978-1-59593-987-6/ 08/ 0001 \$5.00

memory enhancers they require that the user repeatedly construct, consult, and interpret to-do entries. Norman clearly views these as onerous additional tasks that do not help the user with their original task entered into the list.

The focus of research to date on to-do lists has been on best use practices, problems, and desiderata [2,1,7]. We see an immense and largely unexplored opportunity for intelligent assistance in automatically interpreting, managing, automating, and in general assisting users with their to-do lists. This opportunity presents important challenges to intelligent user interface research.

Our initial work on providing intelligent assistance for to-do list management is part of a larger project to develop intelligent assistants for office-related tasks (www.sri.com/calio). As part of this project, a plan execution engine is available to us to act upon and to monitor tasks for users [11] as well as a calendar management system [3] and an instrumented desktop [5]. An important and novel aspect of this project is that the system does not have a pre-defined set of tasks that it can automate; rather, it is continuously learning autonomously and from users (e.g., [13,10,4]). Therefore, users would expect the to-do list manager to handle new tasks as they are learned by the system. CALO includes an execution and monitoring system that can be invoked through a user interface called TOWEL [11]. TOWEL allows the user to jot to-do lists, but does not take action on those to-dos. TOWEL allows the user to invoke agents to perform a task for them. To do that, it generates a form based on the description of the task in the Task Ontology, and the user fills the form to specify the arguments of the task. TOWEL has no capability to automatically interpret the to-do entries to select or fill these forms. Our goal is to provide this capability.

We have discussed elsewhere that an intelligent to-do system can provide a variety of assistance to users [6], not only by acting on to-do list items, but also in monitoring and notifying user on progress on to-do items, augmenting the to-do list with related items, grouping and prioritizing to-do entries, and coordinating to-do lists across users in an organization.

We also discussed some of the challenges of interpreting to-do entries, over and above existing approaches in speech and dialogue systems that assist users with task-oriented

activities [6]. To-do entries are not input with the sequencing or temporal dependencies found in a dialogue system, and therefore must be considered individually and interpreted before the system can infer any potential relationships to other to-do entries. To-do entries are often incomplete and often lack a verb, which makes them hard to map onto the tasks that the system can automate. In addition, many to-do entries in a list can be outside the realm of the system's automation abilities (such as personal matters and tasks that can only be done by the user), while dialogue systems can assume all user utterances are relevant to the task. In other ways, processing to-do list entries is less challenging than dialogue systems since there is no need to keep track of the dialogue history and overall context, and the to-do entries tend to be brief and therefore there is less need for complex grammatical rules.

In this paper, we present an analysis of a corpus of to-do entries collected from users of the CALO system. We then break down the interpretation of to-do list entries as four related activities that can be independently evaluated and each has added value for the user. We also present a baseline system that has simple natural language processing capabilities, and show that they are insufficient to support the interpretation of to-do entries. We end with a discussion on prospects and planned research.

CORPUS ANALYSIS OF TO-DO LIST ENTRIES

We set up a logging system to collect to-do entries jotted by CALO users. This section discusses our analysis of these entries, which illustrates the challenges in developing interpretation and automation aids for to-do lists.

We collected a corpus of 1200 to-do list entries from a dozen users of the CALO system over a period of several months. These users entered to-do list items through the TOWEL interface, where they could drag URLs and documents to the to-do list or enter textual items. We created a reference corpus of 300 entries each by random selection from the original corpus. The other 900 entries were set aside as a test corpus. We analyzed the reference set and report here salient properties of the to-do lists collected. When providing examples throughout this paper, we created fictional to-do entries to protect the privacy of our users.

Of the 300 entries of the reference corpus, 14% had the potential to be automated by CALO. This seemed to be a relatively small number, so as a comparison point we took an informal poll of how many email messages were sent to project assistants versus to others. We asked two users whose primary means of interaction with their (human) project assistants was email to count what proportion of their outgoing emails over a period of three months was sent to their project assistants. The proportion was 4% and 3% respectively. Of the to-do entries with automation potential, 9% of the entries could be mapped to a target task to send email. 3.5% of all entries could be mapped to a

target task concerning scheduling. The remainder were concerned with planning travel and planning a visit.

The remaining entries of the reference corpus, 86%, could not be automated and seem to serve instead as reminders to the user, such as to-do entries for writing documents, fixing coding bugs, and a number of URLs. In our particular reference set, many had to do with programming tasks, which CALO is not intended to automate. Others used very project-specific terms to refer to documents, or system components, or topics, etc. that would be hard to interpret without more context about the user. In future work, this context could be obtained from CALO's instrumented desktop [5]. Nevertheless, this is an interesting category because it can be used as candidates for the system to extend its capabilities and automate those tasks through learning [13,10,4].

67% of the entries did not begin with a verb. A few additional entries had the head verb stated as a gerund, others had a verb but in the later part of the entry. This makes the interpretation task more challenging, since these entries are harder to anchor to a task by looking and the synonyms of the head verb.

We also found unexpected to-do entry structures, such as entries stated as question statements (2%), entries with abbreviations of tasks (1%), and erroneous entries containing typos or interrupted entries (4%).

Because we were interested in automation, we looked more closely at the entries that could be automated. As we mentioned, these were 14% of the corpus. Of those, 56% did not specify at least one argument of the task. 7% had no arguments at all specified. Of the 44% that had all the arguments specified in some form. When to-do items had arguments specified, the specification was very ambiguous. For example, "Meet with Joe" specifies that the person to meet with is "Joe", but there may be no object in the system with an ID of "Joe" but rather 8 or 9 different people IDs registered in the system whose first name is "Joseph" and of those perhaps only one works closely with the user and with that context the interpretation would be ambiguous. Similarly, an entry "Discuss requirements with group" may prompt a task to schedule a meeting, but what the user means by "group" is not simply a language processing task because it also requires context. Some entries referred to a person indirectly, for example "Tell John Smith's boss about the purchasing delay". Other entries referred to specific groups of people using the name of their institution, such as "Tell ISI folks about the new TOWEL features". None of the to-do entries contained enough information to perform a mapping to the arguments without using information about the user context.

Of these entries that could be automated, only 7% had no verb. The rest all started with a verb, though one had an abbreviation of a verb ("sched wed 15th ISI"). This is very disproportionate compared to the 67% of the total entries. However, we hypothesize that verbs are easier to state when

tasks are more concrete and could be automated, as it was with this subset. For tasks that are more abstract and longer-lived, such as “IUI paper”, many activities may be involved and verbs may be less appropriate or even more challenging to come by while jotting the entry.

This corpus analysis highlights some of the challenges that we face in interpreting and automating to-do entries.

INTERPRETING TO-DO LISTS

Interpreting to-do entries involves mapping the user statement in natural language into the formal advertisement of the capability of some agent in the system. For example, a to-do entry stated by the user as:

“Set up discussion on IUI paper”

could be automated by a calendar scheduling agent. The advertisement of this capability may be advertised internally in the system as:

```
<Task name="ScheduleMeeting">
  <agent>CalendarAgent</agent>
  <input-arguments>
    <arg-p type="attendees"/>
    <arg-t type="timeframe"/>
    <arg-s type="topic"/>
  </input-arguments>
  <output-arguments>
    <arg-m
      type="calendar-entry"/>
  </output-arguments>
</Task>
```

which we will refer to in abbreviated form as:

ScheduleMeeting +person +timeframe +topic

Note that we assume that each task has a name and a set of arguments. The arguments can be input arguments, which must be provided in order for the agent to automate the task, and output arguments, which are values that the agent returns upon completion of the task. The system internally manages additional information that is not reflected in these advertised capabilities, such as detect failure conditions, invoke other agents to perform subtasks or to negotiate accommodations (eg changes to an already fully-scheduled day to accommodate a hard to schedule meeting), and request specific interventions from the user.

Interpreting a to-do entry involves four activities:

- 1) **Identification:** Identifying to-do entries that can be mapped to tasks represented in the Task Ontology
- 2) **Selection:** Selecting one or more tasks in the Task Ontology that could be used to automate the to-do entry
- 3) **Association:** Associating portions of the to-do-list entry that can be mapped to arguments of a given task in the Task Ontology
- 4) **Interpretation:** Interpreting the to-do list entry by creating formal objects for the arguments

Identification involves determining whether a to-do entry is within the scope of what the system can automate, i.e.,

whether the agents have advertised capabilities that are relevant to accomplishing the user’s entry. Some examples of successful identification include

“Set up discussion on IUI paper” -> Yes
“Set up salary review with John” -> Yes
“Set up doctor’s appointment” -> No
“Meet Paul on Friday” -> Yes
“Meet UCLA visitors on Friday” -> Yes

Notice that for this task the system does not need to be able to discriminate among tasks in the Task Ontology, only detect that there is at least one relevant task. For example, the last item may map to a ScheduleMeeting task or to a RescheduleMeeting task if there is already a meeting with Paul scheduled before that Friday.

Selection involves determining which task in the Task Ontology is an appropriate capability for the system to accomplish the to-do entry. For example:

“Set up discussion on IUI paper” -> ScheduleMeeting
“Set up salary review with John” -> ScheduleMeeting
“Set up doctor’s appointment” -> NULL
“Meet Paul” -> ScheduleMeeting

Association involves mapping subportions of the to-do entry to appropriate arguments specified for the task in the Task Ontology when appropriate. For example:

“Set up discussion on IUI paper”
 -> ScheduleMeeting
 -> topic = “IUI paper”
“Set up discussion with John”
 -> ScheduleMeeting
 -> attendees = “John”

Interpretation involves identifying objects in the ontology that correspond to the entities mentioned in the to-do entry. Some of the arguments of the task may not have any corresponding objects in the ontology and are left as they were associated to the task representation. For example:

“Set up IUI paper discussion with Jon Smith”
 -> ScheduleMeeting
 -> attendees = calo:Jonathan-F-Smith
 -> topic = “IUI paper discussion”

Notice that for each of these four activities the system could generate more than one candidate answer, as well as some indication of confidence. The four activities need not be separate and may be interleaved. The user would ultimately be presented with a set of interpretation choices for a given to-do entry. For example:

Suggestion #1:
Mapping: scheduleMeeting +person +timeperiod +topic
Bindings: [<topic> = “IUI paper”]
Confidence: .89
Suggestion #2:
Mapping: scheduleMeeting +person +timeperiod +topic
Bindings: [<person> = “IUI paper”]
Confidence: .67
Suggestion #3:
Mapping: rescheduleMeeting +person +timeperiod +topic
Bindings: [<topic> = “IUI paper”]
Confidence: .20
Suggestion #4:
Mapping: schedule +visitor

Bindings: [<visitor> = "IUI paper"]
Confidence: .05

Ideally the user would always get the correct choice shown to him or her in the interface, and therefore of particular interest is the precision and recall for the top choice presented to the user. By default the system should show the user only the highest ranked suggestion, only show the next k highest ranked suggestions upon user request. A user may look at a small number of options (anywhere from k=3 to k=10 may be a manageable set depending on the user's gain) in training/learning mode to provide feedback to CALO so the system can improve its performance overtime. But for normal use, ideally only the top suggestion (k=1) would be seen by the user.

It is important to note that good performance in identification and selection is already useful to the user. Identification enables the system to highlight the tasks that are relevant to the office assistant, so the user can then select the appropriate form and fill it up. Selection enables the system to go one step further and suggest the appropriate form to be filled. Association would enable the system to fill some of the arguments automatically. Interpretation would result in more arguments being automatically filled. But assisting with to-do lists does not stop here. The system would want to be capable of Completion to fill the whole form automatically, by exploiting the user's context and history to learn to make educated guesses about how the user needs or prefers tasks to be accomplished. In addition, the system would ideally decide on Invocation, by making an educated guess on whether it should confirm with the user whether to go ahead and perform the task or simply proceed without consulting the user.

CONCLUSIONS

Assisting users with to-do lists is a new and challenging research area for intelligent user interfaces. In this paper, we presented an analysis of a large corpus of to-do entries with respect to the potential and challenges for automation of those entries. We also presented a breakdown of the interpretation process of a to-do entry into four related but independent stages. Even if full interpretation is not achieved, a system for to-do list management that addresses some of these stages may provide useful assistance to a user in managing to-do lists.

ACKNOWLEDGMENTS

We gratefully acknowledge funding for this work by DARPA under contract no. NBCHD030010. We thank Tim Chklovski for many useful discussions in the early stages of this work. We also thank Karen Myers and Ken Conley for their feedback on this work and its integration with CALO.

REFERENCES

1. Bellotti, V.; Ducheneaut, N.; Howard, M., Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. *ACM Conference on Human Factors in Computing Systems (CHI)*, 2003.
2. Bellotti, V., B. Dalal, N. Good, P. Flynn, D. Bobrow, N. Ducheneaut. What a To-do: Studies of Task Management Towards the Design of a Personal Task List Manager. *ACM Conference on Human Factors in Computing Systems (CHI)*, 2004.
3. Berry, P.; Conley, K.; Gervasio, M.; Peintner, B.; Uribe, T.; and Yorke-Smith, N. Deploying a Personalized Time Management Agent. *Proceedings of AAMAS'06 Industrial Track*, Hakodate, Japan, May 2006.
4. Blythe, J. Task Learning by Instruction in Tailor. *In Proc. of Intelligent User Interfaces (IUI-2005)*, 2005
5. Cheyer, A. and Park, J. and Giuli, R. IRIS: Integrate. Relate. Infer. Share. *1st Workshop on The Semantic Desktop. 4th Intl Semantic Web Conference*. 2005.
6. Gil, Y. and Chklovski, T. Enhancing Interaction with To-Do Lists Using Artificial Assistants. *AAAI Spring Symposium on Interaction Challenges for Artificial Assistants*, Stanford, CA, March 26-28, 2007.
7. Hayes, G., Pierce, J. S., and Abowd, G. D. Practices for Capturing Short Important Thoughts. *ACM Conference on Human Factors in Computing Systems (CHI)*, 2003.
8. Jones, S. R. and P. J. Thomas. "Empirical assessment of individuals personal information management systems." *Behaviour & Information Technology* 16(3), 1997.
9. Kirsh, D. "The Context of Work", *Human Computer Interaction*, Vol 16, 2001.
10. Mitchell, T., Wang, S., Huang, Y., and Cheyer, A. Extracting Knowledge about Users' Activities from Raw Workstation Contents. *The Twenty-First National Conference on Artificial Intelligence (AAAI '06)*, 2006.
11. Myers, K., P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, and M. Tambe. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine*, 28(2), 2007.
12. Norman, D. A. Cognitive artifacts. In J. Carroll, editor, *Designing Interaction: Psychology at the Human-Computer Interface*, pages 17--38. Cambridge University Press, New York, NY, USA, 1991.
13. Shen, J., Li, L., Dietterich, T., Herlocker, J. (2006). A Hybrid Learning System for Recognizing User Tasks from Desktop Activities and Email Messages. In *2006 International Conference on Intelligent User Interfaces (IUI)*. 86-92. Sydney, Australia.