To appear in ACM Transactions on Interactive Intelligent Systems (TiiS), 2012.

Capturing Common Knowledge about Tasks: Intelligent Assistance for To Do Lists

Yolanda Gil
University of Southern California

Varun Ratnakar
University of Southern California

Timothy Chklovski
Factual Inc.

Paul Groth
Free University of Amsterdam

Denny Vrandečić
Karlsruhe Institute of Technology

Abstract

Although to-do lists are a ubiquitous form of personal task management, there has been no work on intelligent assistance to automate, elaborate, or coordinate a user's to-dos. Our research focuses on three aspects of intelligent assistance for to-dos. We investigated the use of intelligent agents to automate to-dos in an office setting. We collected a large corpus from users, and developed a paraphrase-based approach to matching agent capabilities with to-dos. We also investigated to-dos for personal tasks, and the kinds of assistance that can be offered to users by elaborating them based on sub-step knowledge extracted from the Web. Finally, we explored coordination of user tasks with other users through a to-do management application deployed in a popular social networking site. We discuss the emergence of Social Task Networks, which link users tasks to their social network as well as to relevant resources on the Web. We show the benefits of using common sense knowledge to interpret and elaborate to-dos. Conversely, we also show that to-do lists are a valuable way to create repositories of common sense knowledge about tasks.

Contact author: Yolanda Gil, Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Marina del Rey CA 90292, gil@isi.edu.

1. 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 [Kirsh 2001]. 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 [Jones and Thomas 1997].

There have been user studies of to-do lists that have uncovered use patterns and design desiderata (e.g., [Bellotti et al 2004; Hayes et al 2003]). Many to-do list managers are widely available online (e.g., todolistsoft.com, tadalist.com, todoist.com, voo2do.com among many others). These types of systems help users by storing to-dos, and provide very simple assistance by synchronizing to-dos between devices, providing reminders, creating repeated to-do items, etc. However, no system has been developed to date to provide intelligent assistance to users. As [Norman 1991], we see an immense and largely unexplored opportunity through automatically interpreting, managing, executing, and in general assisting users with to-do lists. To-do lists include items that can be acted upon by intelligent agents that have the capabilities to accomplish some of those items. This kind of investigation becomes possible as agent-based systems are developed to assist users with routine tasks, such as office assistants (e.g., Electric Elves [Chalupsky et al 2002] CALO [Myers et al 2007; Berry et al 2006], SIRI [Cheyer et al 2005]) and other technologies that exploit agents and services on the web (e.g., [Berners-Lee et al 2001]). The social web offers new opportunities for coordination of to-dos across users. These opportunities present important challenges to intelligent user interface research. Assisting with to-do lists requires natural language (NL) processing and understanding, a significant amount of common sense knowledge, appropriate user interaction strategies, and coordination support.

This paper describes our work in exploring different kinds of intelligent assistance for To-do lists. The paper reports on data collected from users of progressively more sophisticated versions of our software. The data collected and the evaluations performed were meant to inform our research, rather than prove conclusively the significance of user behaviors.

The focus of our initial work was on mapping to-dos into the capabilities of agents that can automate those entries for the user [Gil and Ratnakar 2008a; Gil and Ratnakar 2008b]. Our journey began by collecting To-do data in a controlled setting where users were performing office tasks. After understanding the particular characteristics of To-do lists, we designed an approach to interpret them and map them to automated capabilities of agents available in the system. Our approach used paraphrase patterns to map to-do entries into formal task descriptions representing agent capabilities. We then moved to a setting where users were entering to-do lists for personal tasks [Vrandečić et al 2011]. We explored the use of web repositories of task knowledge to assist users in elaborating and decomposing to-dos. These experiences led us to investigate to-dos in more collaborative settings. We developed a to-do system for managing to-dos in a social networking site, where users could share to-dos and know-how within their social network [Gil et al 2009; Gil et al 2010]. We arrived at the notion of Social Task Networks, which intertwine user tasks among themselves, with the user's social network, and with web resources to create a rich tapestry of what personal tasks are all about.

Table I. Challenges for Assistance with To-Do Lists

Natural Language Interpretation
To-dos can be: • Minimal or cryptic • Higher level than automatable tasks • Irrelevant or cryptic • No action is preferable

Common Sense Knowledge
Useful knowledge about tasks: • Substeps • Paraphrases of tasks • Repairs • Ordering of subtasks • Duration

User Interaction
Assistance can be: • Act on to-do item • Monitor to-do list • Elaborate to-do list • Organize to-do list entries • Prioritize to-do entries

Coordination
Coordinating to-dos with: • Ongoing user activities • Other personal information tools • Other users' tasks • Organization processes and practices

The paper begins with an overview of challenges for assisting users with to-dos. Then it presents our work on to-do interpretation in the context of an agent-based office assistant system. We then contrast office to-dos with another corpus that we collected of personal to-dos. Finally, we discuss collaboration issues in to-do list management in the context of a to-do management system that we developed for a popular social networking site. We introduce Social Task Networks, and our work on publishing user tasks as open shared Semantic Web objects.

2. CHALLENGES FOR ASSISTANCE WITH TO-DO LISTS

The challenges in assisting users with to-do lists are many. We describe them here along four dimensions: 1) natural language interpretation of to-do items, 2) user interaction design, 3) coordination with other activities and other users, 4) awareness of context and common sense knowledge about tasks. An overview and summary is shown in Table 1.

2.1 Interpreting Free-Text To-Do Items

To-do list entries are not well-formed or complete sentences typically found in text, and lack the dialogue context that is present in conversational interfaces (e.g., [Allen et al 2001]). Nevertheless, the task of interpreting the user entries has relevant research in speech systems and dialogue natural language interfaces which allow users to query databases or assist them with performing tasks [Seneff, 1992; Glass et al, 2004; Allen et al, 1995, 1996, 2001]. These systems must also recognize and track the progress of a task during successive interactions with the user, and be flexible regarding which tasks the user undertakes and which tasks are left to the system [Rich & Sidner 1997, Allen et al 1995, Rudnicky et al 1999]. As in those systems, interpreting to-do list entries involves mapping the surface text used in the user's utterance into the set of tasks for which the system has automated procedures. Typically, the approach taken is to pre-enumerate the set of tasks that the system can perform, to collect a corpus of utterances for those tasks, and to develop a grammar that is used to map user utterances into the internal task representation. This mapping grammar is typically developed by hand, although recent research begins to address the issue of learning some aspects of these

grammars and mappings automatically [Wang & Acero 2001, 2002; Chung et al, 2005]. Another approach that we have adopted is to collect paraphrases from users to extend the set of statements that can be interpreted [Gavalda & Waibel, 1998; Chklovski 2005].

However, interpreting to-do list entries presents important new challenges.

To-do list entries are **often very minimal or cryptic** (e.g., "Jack"). In these cases their mapping is more challenging and their intent has to be inferred. In our work, since it is in initial stages, we assume that to-do list entries are specified as an action followed by some optional arguments and constraints, much in the spirit of a case frame [Fillmore 1969]. The mapping and interpretation of the entry is still challenging, since the specification of the action and the arguments are not fixed and can have variations: "form /make/set up/pull together/get together/identify/id a lunch group", and "hotel, place to stay, hotel room, accommodation, Marriott, motel". We believe this is a reasonable assumption based on an analysis of realistic to-do data (public Ta Da lists, www.tadalist.com/lists/select) and people's goal statements (on www.43things.com). Although we acknowledge that to-do entries are sometimes stated more cryptically, we observed that they often conform to our assumption. It is possible that users may adopt our required format, since users commonly adapt after interacting with computers or other humans that use different vocabularies and languages than their own [Zoltan-Ford 1991; Ringle and Halstead-Nussloch 1989]. We provide a mechanism in the interface to allow users to see what portions of their to-do list entries are understood by the system. If our assumption is not desirable for users, our plan is to develop techniques that learn to manage minimally specified to-do entries after watching user abbreviation practices for some period of time.

Another important challenge is that to-do entries may be **stated at a much higher level than the task that may be known to the system**. For example, the entry may be "Define project goals with Jack" and the system may only be able to help with the task of setting up a meeting with that person. In speech systems and other NL-based interfaces the user's utterances always need to be mapped to a pre-defined set of tasks (travel arrangements, phone menu navigation, etc). In our case, the task that the system knows how to-do may be very small and tangential the to-do entry itself, but it is very necessary, and burdensome, and is so totally obvious and repetitive that the user would expect an intelligent assistant to be able to anticipate and handle. How can the system figure out that defining project goals with a person involves setting up a meeting?

An important new challenge is that to-do lists will contain entries that are **not relevant to the system's intended assistance**. The system should be able to determine which entries are outside of its known responsibility areas and with which it cannot assist the user. In speech systems and other NL-based interfaces the user's utterances are relevant to the task at hand, a reasonable assumption given the context of their use. To-do lists present an additional challenge for users that naturally keep a single to-do list that does not separate their personal from their office tasks. For example, the list may contain an entry to-do a budget or to get bread on the way home, which is not the kind of task that the system is concerned with. How can the system figure out that buying bread on the way home is not a task one does at the office?

Another challenge is that users will **prefer no action than incorrect or burdensome assistance**. Speech and NL systems must try to interpret users' utterances before the dialogue may proceed. User's expectations for to-do list assistants, at least at this point in time, will be that they help when possible but are not required to assist with every to-do entry thoroughly. If the system cannot interpret an entry, or if it believes its interpretation or mappings of the entry are not

likely correct, then it should give up on that entry and not attempt any assistance. How will the system determine when to give up trying to assist the user with a to-do entry?

Finally, a challenge we face is that the underlying **system that will be continuously expanding its capabilities** in terms of the kinds of tasks it can do for the user. Speech systems and other NL-based interfaces target a pre-defined set of tasks. But with to-do lists, the set of tasks is ever expanding. If the system is extended (or learns on its own) to RSVP for events through the Internet, how can it identify occurrences of event invitations or related tasks in a to-do list?

In summary, new challenges involved in developing an approach to interpret to-do list items include: 1) anticipate the preparatory or minor tasks which are involved in accomplishing the user's stated task; 2) determine which tasks are proper to an office environment and which are within the realm of the system's assistance and which are not; 3) determine when automation is desirable; and 4) add and extend the mappings between user utterances and new internal task representations learned by the system over time.

2.2 User Interaction

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 [Norman 1991] 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 memory enhancers they require that the user repeatedly construct, consult, and interpret to-do entries. Norman rightly views these as onerous additional tasks that do not help the user with their original task entered into the list.

Clearly, any intelligence assistant for to-do lists should be proactive and provide not just assistance with to-dos but support a variety of related activities such as:

- **Act on the to-do list.** This entails mapping entries to the automated tasks that the system can carry out on behalf of the user. To launch the appropriate automated task, the system needs to both correctly identify the appropriate task and bind the relevant arguments to the parameters of its internal task description. To-do list entries, however, can be notoriously incomplete, since users may simply jot them as reminders. Therefore the mapping problem is challenging since it will typically be underconstrained.
- **Monitor the to-do list.** An entry on the list may become completed or cancelled: for instance, a conference room booking may get automatically carried out, or a planned interview may get cancelled. In such cases, the system can monitor update the to-do list, marking its entries as completed or canceled as appropriate.
- **Elaborate the to-do list.** Given an entry, the system could suggest additional preparatory tasks and sub-tasks that are not on the to-do list. For instance, given a to-do entry "host a visitor", the system could suggest the substep of reserving a hotel room for the visitor, for which automated assistance can be provided to the user. Another form of augmentation is the extension of individual entries with additional information, eg, the visit's date, if such is available in the larger system.
- **Organize to-do list entries.** The system could organize the list and group entries by different activity threads, organizing and grouping the tasks entered by the user as well as the tasks added by the system. For example, the system may group the items that are related to hosting a specific visitor.

- **Prioritize to-do list entries.** Over time, tasks that have an associated date or deadline become more pressing. In addition, there may be time constraints on the advance execution of preparatory tasks, which if not accomplished would delay the to-do list entry target date or altogether render it impossible.

Another important aspect is the design of cognitively appropriate interactions and presentations for to-do lists that make users more effective at their tasks and even at introducing better habits. This aspect is not addressed in our work, but has been the focus of user studies of to-do list best use practices, problems, and desiderata [Bellotti et al 2004; Bellotti et al 2003; Hayes et al 2003].

In summary, the user interaction with an intelligent assistant for to-dos should not be limited to facilitating the accomplishment of to-dos but should also include identifying assistance opportunities on its own, managing and maintaining the to-do list, and anticipating possible tasks which the user may not have mentioned.

2.3 Incorporating Common Sense Knowledge About Tasks

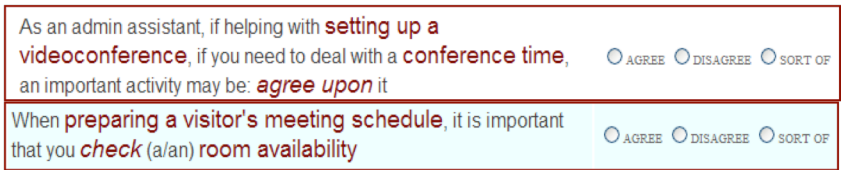
There is a significant amount of common sense knowledge about tasks that could be brought to bear in making assumptions about how to interpret and assist users with to-do lists. Common sense knowledge has been used in a variety of contexts to assist users with tasks such as organizing pictures [Lieberman et al 04] and personal task management [Smith and Lieberman 2010].

In prior work, we have created several repositories of broad knowledge about tasks that could be used to improve interpretation and assistance of to-dos. We obtain common knowledge about tasks from two main sources: volunteer contributors over the web (or within the organization), and text extraction from on-line corpora. We developed techniques to guide collection from volunteers, to expand coverage, to process and relate the contributed statements, and to validate the knowledge collected [Chklovski & Gil, 2005; Chklovski 2003, 2005]. We also extracted task information from a variety of corpora Chklovski & Pantel 2004, 2005].

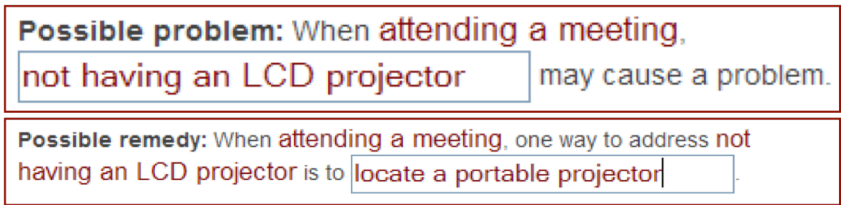
The types of relevant information and the kinds of repositories that we created include:

- **Objects relevant to a task.** The Repairs Repository contains thousands of statements about objects, their typical uses, and their parts, as well as a smaller set of statements about repairing difficulties one may encounter in the office environment, such as "if projector is not working, try a new bulb" [Chklovski & Gil, 2005]. Such knowledge can be of help in interpreting to-do entries which refer to problems, and relate tasks to objects or their parts.
- **Descriptions of tasks in natural language.** Our Action Paraphrases Repository contains thousands of paraphrases collected from volunteers [Chklovski 2005; Chklovski & Gil 2005]; the repository includes several hundred paraphrases which were specifically collected to target office- and assistant-related tasks, including: "plan a visit ⇔ schedule a visit", "arrange a meeting ⇔ schedule a meeting", "lease a car ⇔ rent a car".
- **Decompositions of tasks into subtasks.** Our Substeps Repository contains knowledge such as "attending a conference has-substep registering for the conference", and "go to meeting has-substep find location of meeting", and the sometimes true "have lunch has-substep buy food". This repository is based on data collected by OMCS [Singh et al, 2002] and Learner [Chklovski, 2003].

(a) Objects relevant to tasks



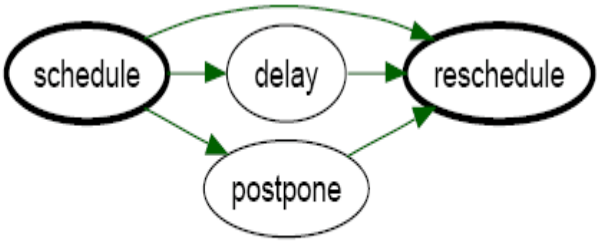
(b) Subtasks



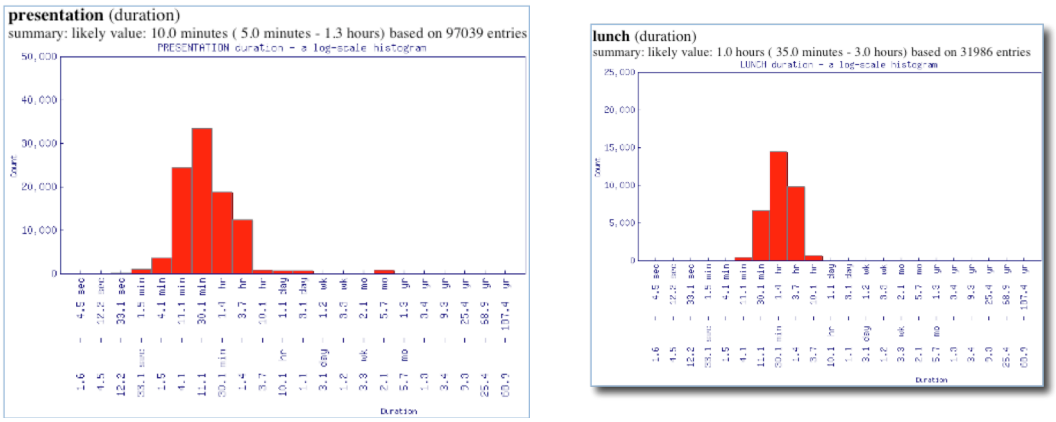
(c) Potential failures and repairs

(d) Paraphrases

Figure 1. Collecting Common Knowledge about Tasks from Volunteers.



(a) Relationships between taks



(b) Durations of tasks

Figure 2. Common Knowledge about Tasks Extracted from Textual Corpora.

- **Relations between tasks.** Our Verb Relations repository, VerbOcean [Chklovski & Pantel 2004, 2005], was extracted from the Web. It contains more than 22,000 instances of relations between nearly 3,500 distinct verbs,including "schedule happens-before reschedule", and "cancel stronger-than postpone". Such relations can be useful in identifying potentially related to-do entries and their subtasks.
- **Durations of tasks.** Our Task Duration repository is extracted based on frequency of temporal mentions in the Google TeraWord corpus [Brants and Franz 2006]. This is an English word 5-gram corpus that was made widely available, extracted from trillion-word tokens of text in public Web sites.

Figure 1 shows snapshots of the interfaces to collect different kinds of common sense knowledge about tasks from volunteers. Figure 2 shows examples of task knowledge extracted automatically from text.

These repositories have broad coverage of the topics and tasks of relevance, since they include knowledge about common tasks and in some cases have more dense coverage of office tasks in particular. The repositories are semi-formal because they

are formed by normalizing and relating English statements provided by web volunteers or that appear in text documents.

2.4 Coordination with Other Activities and Other Users

Tasks are seldom self-standing and isolated from the rest of a user's interactions with the world. Most tasks require significant coordination, and a to-do assistant should have awareness of a variety of dimensions for coordination:

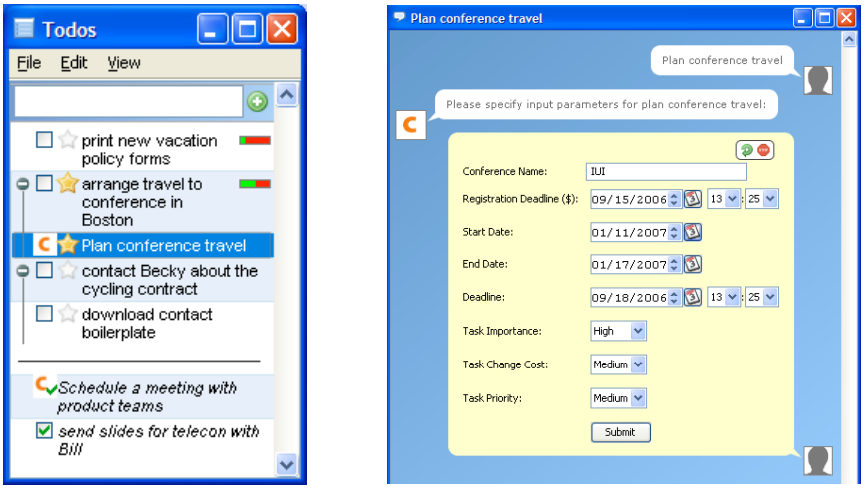
- **Coordinating to-dos with ongoing user activities.** Any intelligent assistant should be sensitive to the cost of interventions and interruptions to the user, the added value of the interaction, and be cognizant of the tradeoff space of adjusted autonomy.
- **Coordinating to-do lists with other personal information management tools.** Many tasks manifest themselves in a variety of interaction channels for the user. To-do lists are just one of those channels, and need to be coordinated with aspects of the task that appear on emails, calendars, and other notes. Moreover, many aspects of a task may be implicit, for example on the locations they frequent or on the set of active documents that a user may be editing.
- **Coordinating tasks with other users.** Many user tasks are collaborative, presenting opportunities for coordination. For example, if the user meets regularly with several others, it is possible that one of them usually takes the initiative to set up the meeting time and therefore the user does not have to.
- **Coordinating with organization processes and practices.** This aspect of assistance is to help new users understand practices within the organization, or help users keep up with changes in practices. For instance, taking time off may require filing specific paperwork, or travel arrangements may be delegated to project assistants, depending on the particular organization. A system for interpreting to-do lists may be able to advise the user and act accordingly.

Several systems have focused on providing task assistance to users based on the content of their email and calendar [Shen et al. 2006; Freed et al 2008]. These systems take advantage of both the content of task artifacts but also their structure [Kushmerik and Lau 2005]. For example, a system may analyze the text of an email but will also derive information from how the text is organized into sender, receiver, subject, and body fields. This extracted structured information can facilitate the coordination with to-do lists management tools and intelligent assistance.

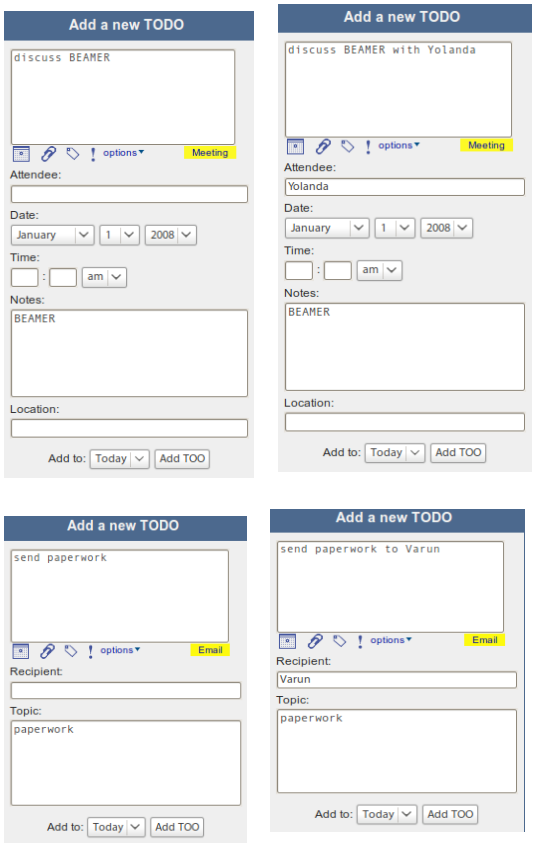
In summary, intelligent assistants for to-dos should provide support to coordinate with other user tasks, other personal information management tools, with other individuals, and with organizational processes.

3. INTERPRETING TO-DOS : EXPERIENCES WITH TO-DOS IN A WORK ENVIRONMENT

A central part of office work is the collection, assignment, sharing, tracking and scheduling of tasks. Such task management activities are performed using a variety of tools, most commonly to-do lists, calendars and email.



(b) In the CALO TOWEL interface, BEAMER identified when an agent could assist with the to-do, and marked the to-do to notify the user, who could then task the agent.



(b) In the RADAR interface, BEAMER identified an agent that could assist with the to-do, and started to fill out the agent's tasking interface.

Figure 3. User interfaces for to-do list management to task agents. Our BEAMER to-do interpretation system was integrated with two different agent-based office environments.

Our initial work on providing intelligent assistance for to-do list management was part of a larger comprehensive project to develop intelligent assistants for office-related tasks (http://pal.sri.com/, http://ai.sri.com/calo). As part of this project, a plan execution engine is available to us to act upon and to monitor tasks for users [Myers et al 2007] as well as a calendar management system [Berry et al 2006] and an instrumented desktop [Cheyer et al 2005]. 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., [Shen et al 2005; Mitchell et al 2006; Blythe 2005]). Therefore, users would expect the to-do list manager to handle any new tasks learned by the system. The system was tested and evaluated by users in a real office setting at selected intervals throughout the duration of the project.

Figure 3 illustrates the user interfaces for two different office assistants. Figure 3(a) shows the user interface for TOWEL [Conley and Carpenter 2007; Myers et al 2007], where the letter "C" indicated that the to-do could be automated by an agent. Figure 3(b) shows the user interface for another office assistant: the RADAR system [Freed et al 2008]. RADAR is an automated personal assistant with extensive capabilities that include processing email through extraction, management, and execution of email prompted tasks. RADAR has a capability to extract tasks from email messages. BEAMER provides a complementary capability to extract tasks from to-do lists. BEAMER takes to-do entries as a user types them in the Radar to-do entry interface. Beamer first selects a target agent task that is the highest ranked match for the to-do entry. Then it takes the form corresponding to the target task and fills it with information taken from the to-do entry.

Even outside of an office assistant system, automation could be possible. Some tasks could be automated by on-line scripting applications, such as IBM's Co-Scripter, iMacros for Mozilla's Firefox, or Yahoo Pipes. The user interface for to-dos should be unchanged, even though the underlying execution engine is widely different for each of these systems. The to-do interface should just allow the seamless integration with these different execution engines.

Our work focused on interpretation of to-dos. That is, in bridging the gap between the actionable tasks that the system could perform and the informal to-do lists of users, and in learning to map new tasks as the underlying execution system learns them. Our implemented system, BEAMER, had only an initial set of the to-do list management capabilities but it was fully integrated with an end-to-end execution and monitoring system. We were able to collect a sizeable corpus of real to-dos in an office setting and access a number of agents with relevant capabilities to automate to-dos, which enabled us to carry out a quantitative evaluation of our approach.

3.1 Corpus Analysis of Office To-Dos: Opportunities for Automation

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

We collected a corpus of 2400 to-do list entries from a dozen users of the system over a period of several months. These users entered to-do list items through the TOWEL interface, shown in Figure 2(a), where they could drag URLs and documents to the to-do list or enter textual items. The to-do entries we are using for our evaluation were collected with no automatic support when being entered (like auto-completion or suggestions). All users were part of the same organization and therefore the same office environment.

We created a reference corpus of 300 entries each by random selection from the original corpus. The other 2100 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 agents. 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 the overall system was 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. This context could be obtained from an instrumented desktop that was developed as part of the overall project [Cheyer et al 2005]. 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.

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, 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 features". None of the to-do entries contained enough information to perform a mapping to the arguments without using information about the user context.

Table II. Summary of findings from corpus analysis of office to-dos.

67% did not begin with a verb
2% were stated as questions
4% contained typos
14% could be automated by some agent in the system. None could be directly automated because: • 56% we missing at least 1 argument for the task   o 7% had no arguments • 44% had all arguments but were ambiguous In addition: • 93% had a verb

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.

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

3.2 Interpretation of To-Dos for Selecting and Tasking Agents

We considered six phases in interpreting and automating to-do list entries, shown in Figure 3. The left side shows the kind of result expected from each step, and the right side gives an example. As we will see, the automation of each of these steps can be turned into an interface extension that can be useful to the user in its own right.

Relevance involves determining whether a to-do list entry is relevant to any of the agent capabilities. That is, whether the to-do entry is within the scope of the tasks that the system can automate for the user. If the system has good performance in this step, it can help the user by highlighting those entries in the to-do list so that the user is aware that there are agents that could automate them.

Selection involves finding the agent capability (or target task) that is appropriate for automating a given to-do entry. Note that the system may generate several candidates. If the system has good performance in this step, the top ranked candidate can be shown to the user as a possible way to automate the entry. Other candidates, and there should not be too many, can be shown as alternatives to the user as a pull-down menu.

Association will determine which chunks of the to-do entry text correspond to which arguments in the target task. Good performance in this step would allow the system to have the menu to task the agent pre-filled by those mapped arguments, requiring little or no editing from the user to task the agent.

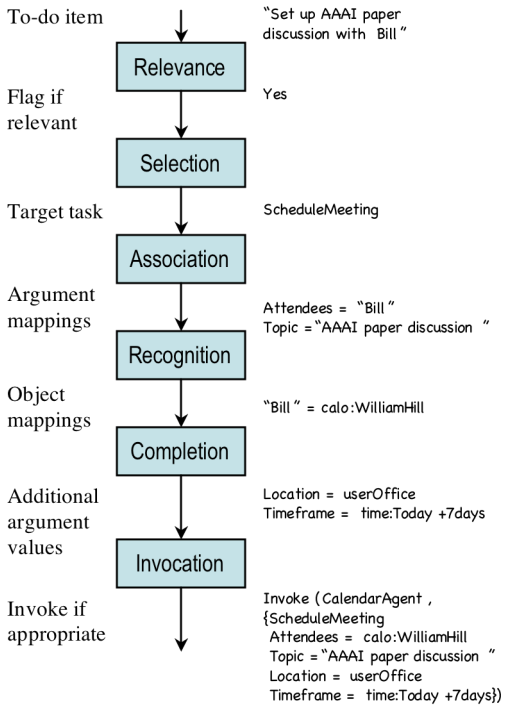


Figure 3: Phases involved in interpreting and automating to-do list entries.

The last three phases are shown here for completion but have not been the focus of our work to date. **Recognition** finds an object in the system that corresponds to the text chunk of an argument. For example, the to-do entry may contain the first name of a person and the recognition phase would find that person's formal representation, which will allow the agent internally to access email addresses or other information useful for the task. Notice that not all arguments can be mapped to formal objects. An example is the topic of a meeting, which is only useful (at least in our work) to the other human attendees. **Completion** involves completing the other argument values not included in the to-do list entry, either by using default values, typical values learned for each user, or values inferred from the user's context (e.g., other meetings, emails, etc.) **Invocation** involves deciding whether the agent should be invoked, which can be done with the user's explicit consent or using some measure of adjustable autonomy for the task.
These last three stages could be done by the agent, where the agent could have mechanisms to set default values and to make decisions about when to act. The first three stages are properly the responsibility of a to-do list manager and are the focus of this work.

3.3 Using Paraphrases to Interpret To-Do List Entries

Our approach to interpreting to-do lists is to use paraphrases to re-state the to-do list entry until it can be mapped to a target task or the system runs out of paraphrases to try. We pursued this direction for two main reasons. Because the to-do entries are incomplete and not well-formed sentences, they do not lend themselves to natural language techniques such as chunking and parsing. The second reason is that we want to improve the system's performance by learning as users use it.

In our approach, the system maintains a list of paraphrase patterns for each target task. A paraphrase essentially reflects how the user might invoke certain tasks. Paraphrase patterns contain special keywords combined with task arguments. Some example paraphrase patterns for a task are:

Task: SendEmail +recipient-s +topic
Paraphrases:
ask +recipient-s to +topic
message to +recipient-s
send +topic to +recipient-s
email +recipient-s about +topic

In this example, *"SendEmail"* is the target task, which has the two arguments *"recipients"* and *"topic"*. Task arguments are marked with a "+" in the paraphrase pattern. Keywords in the patterns include "ask", "to", and "about." The keywords are used for matching paraphrases with the entry, while the arguments are used to map portions of the to-do entry to those arguments. In order to have a successful match, all the keywords in the pattern should match the to-do entry in the order that they are written in the paraphrase. Any portion of text in between matched keywords and marked in the pattern as an argument gets bound to that argument. Each paraphrase pattern has a direct correspondence between its arguments and the arguments in the target task. Lists are marked in the pattern with the suffix "-s" at the end of the argument, which is matched with special keywords such as "and", "or", "/", "," in the to-do entry. Using the paraphrase patterns listed above, these are some example to-do items and their matches:

Ask Bill to give feedback on paper
-> ask +recipient-s="Bill" to +topic="give feedback on paper"
Ask Bill for feedback on paper
-> --No match--
Message to Bill about project report
-> message to +recipient-s="Bill" +topic="project report"
Send message to Bill
-> message to +recipient-s="Bill"

Note that the to-do entry *"Send Message to Bill"* matches the paraphrase pattern *"message to +recipient-s"* as the whole pattern is contained within the entry. In other words, the text matches a paraphrase pattern if all keywords in the pattern match the text. The reverse is not the case, for example the to-do entry *"Email to Bill"* would not be matched with the paraphrase pattern *"send email to +person"* since not all the pattern keywords are present in the to-do entry.

In selecting a target task, there may be several candidates and we produce a ranking based on the match. We define the specificity of the paraphrase as the number of keywords in the paraphrase. In case of a match, the ranking of the match is the specificity of the paraphrase pattern that it matched against.

Paraphrase patterns can be provided by hand, which may be useful to seed the system and bootstrap its initial performance. Our system can learn paraphrase patterns as the user invokes an agent by hand by selecting an agent from a menu and tasking it through a form-filling interface (this is how users task agents with to-do items in the current system). An alternative approach is to learn paraphrases from volunteer contributors where the system can acquire the paraphrases off-line from volunteers by prompting them for variants of already known paraphrase patterns [Chklovski 2005].

To learn paraphrases, we create a variabilized pattern of each to-do entry that is mapped to a target task, where the variables correspond to the arguments in the

target task. Our training examples come from the user as they invoke agents by hand. Other researchers have addressed paraphrase learning from text corpora (e.g., [Barzilay and Lee 2003]), but make use of vast amounts of data (in the order of megabytes) that we do not have available. Learning is currently done by looking at the set of "answers" that are collected from the user over time. These answer keys contain the to-do entry and the matching task and arguments as filled by the user. The system scans through the to-do entry and replaces the text mapped to the argument with the task argument variable to create the paraphrase pattern. An example answer key is:

to-do-entry: email John about demo deadlines
task-type: SendEmail +recipients +topic
arguments: recipients="John" topic="demo deadlines"

which would result in the following learned pattern:

email +recipients about +topic

However, not all paraphrases that are created are eligible for use. We restrict paraphrase patterns as follows. There needs to be at least one keyword, so for example *"+topic"* is not allowed. This restriction is to avoid matching anything and everything. The more specific the paraphrase, the better chance there is of getting a correct match. Another restriction is that arguments must be separated by one or more keywords. For example, *"ask +recipients +topic"* is not allowed. This second restriction is needed because we do not have a recognition phase. That is, the system does not have a mechanism to correctly identify where the first argument stops and the second argument starts. In our current system, that paraphrase pattern would not be allowed (or learned). We discuss this restriction further below.

3.4 Evaluation Metrics for To-Do Interpretation

Since our system is the first of its kind ever developed, we needed to define metrics to measure its performance. Because of the novelty of this research area, we have focused on evaluating the performance of the system's functionality in terms of its internal algorithms rather than on evaluating usability issues. Therefore, we evaluated the system off-line, with a corpus of to-do entries collected from users over several months.

The evaluation tests performance on identification, selection, and association. As we discussed earlier, good performance in each of the tasks can have direct utility to the user.

For task relevance identification, we measure performance with accuracy, precision, and recall metrics. The accuracy metric is defined as the percentage of to-do items that were correctly flagged or not flagged. Precision is defined as the fraction of to-do entries that were flagged as relevant that are actually relevant:

Precision = {to-dos relevant to agent capabilities
AND flagged as relevant}
/ {to-dos flagged as relevant}

Recall is defined as the fraction of to-do entries actually relevant that are actually flagged by the system as relevant:

Recall = {to-dos relevant to agent capabilities
AND flagged as relevant}
/ {to-dos relevant to agent capabilities}

Precision and recall can be combined as a joint F-measure metric that takes their harmonic mean, F-metric = 2/(1/Precision + 1/Recall).

To evaluate performance with respect to task class selection, we want to use metrics that reflect the system's value to the user. Precision and recall would not be appropriate. For example, if an option is generated, but is ranked 17th, the system's precision and recall would be high because the correct selection would have been generated. But since the user may never see it, the performance metrics that we use should reflect that it was not found useful to the user. Therefore, improving on the precision and recall metrics should not be our ultimate goal. Ideally, the system should always aim to choose the right task as its top-ranked suggestion. To measure performance in this respect, we use the following metric:

t-metric = % entries where the correct answer is the top option proposed

To track the system's behavior, we use the following metrics as well:

g-metric = % entries where the correct task mapping is proposed as a candidate
c-metric = average number of candidates generated that are selected to appear in the top k to show user

Note that performance in the g-metric and the c-metric directly affects the t-metric, but our goal is to improve the t-metric performance.

For association, we measure performance in terms of how many corrections would the user have to do to the initial mapping and bindings proposed by the system in order to get the desired argument mappings. That is, a given suggestion will be used to fill the initial task description that the user will correct if needed. Our metric is the *edit distance*, namely how many changes would the user need to make to the system's mapping and bindings if the user had to correct the top suggestion.

3.5 Evaluation Methodology for To-Do Interpretation

For our evaluation, we obtained to-do entries and processed them off-line. These to-do entries were obtained as users were jotting them as part of their normal use of the system. The system was completely passive and provided no assistance with the to-do lists. Our goal at this stage of the project is to assess the quality of the system before it is deployed for use with true on-line learning and assistance.

2.1.1. Evaluation Corpus. We collected a corpus of 2400 to-do list entries from a dozen users during two subsequent annual evaluation periods that lasted several months each. A subset of 300 entries were extracted as a reference corpus and used for development and analysis purposes. The remainder 2100 entries were set aside for this evaluation. When providing examples throughout this paper, we created fictional to-do entries to protect the privacy of our users.

2.1.2. Target Tasks. We used a Task Ontology where agents registered their capabilities. It contained a set of 87 task categories that were used to evaluate the system. These task categories only have the class name, and no arguments are specified. Most of them are not relevant to the to-do list corpus collected. Example task categories are: Decide, Explain, Move, Calculate, Execute, Commit, Learn, Affect, Wait, Input, Borrow, Claim, AnswerWithChoice, AnswerWithData, Announce, and Invoke. We selected a subset of these tasks that we could map reasonably well

to capabilities of agents available in the system. We represented their arguments based on the kind of information that automating these tasks would require for these agents. This resulted in a set of 18 candidate target tasks that we used for evaluation:

PlanTrip +location
PlanTrip +event +date +location
ScheduleVisit +person +timeframe
ScheduleInterview +visitor +timeframe
ScheduleSeminar +person +timeframe
ScheduleMeeting +attendees +topic +timeframe
RescheduleMeeting +attendees +timeframe +topic
ScheduleGroupVisit +group +topic +timeframe
Buy +equipment +person
Buy +office-supplies +person
Buy +flight-reservation +person
Lookup +document
Download +url
Print +document
Edit +document
Approve +purchase-request
Approve +travel-request
SendEmail +recipients +topic

2.1.3. Answer Set. We asked an annotator to label the test set with the correct answers for each task. In some sense only the user that created the to-do item would know what the correct answers were based on their true intention and context, so the annotator took a best guess at what the tasks were intended to accomplish. For example:

To-Do-Entry: Setup discussion on IUI paper
Correct-Selection: ScheduleMeeting +person +timeframe +topic
Correct-Association: topic = "IUI paper"

The to-do entries that did not correspond to any task were given NULL selection and association labels. There were a total of 382 entries that were marked with a target task, the rest did not correspond to any task. For the entries that were mapped to tasks, 130 were mapped to *SendEmail*, 90 to *Lookup*, 58 to *Edit*, 51 to *ScheduleMeeting*, 25 to the *PlanTrip* tasks, 10 to the *Buy* tasks, 9 to *ScheduleInterview*, 3 to *ScheduleVisit*, and 2 each to *Download*, *Print* and *ScheduleSeminar*. There were three tasks that did not appear at all in the annotated reference corpus and were the *RescheduleMeeting*, *ScheduleGroupVisit*, and *Approve* tasks.

Notice that the tasks included in each of the sets are not completely unrelated to one another in that they have many terms and argument types in common. Therefore, the system must discriminate among them correctly.

3.6 Evaluation Results for To-Do Interpretation

We compared the performance of our system with respect to a baseline system that made random decisions, as well as with several configurations of bag of words matching. Our system is shown as "ordered matching," to distinguish it from bag of words matching of paraphrase patterns. We show learning curves for each metric at 8%, 15%, 30%, 60% and 100% of the training data using 4-fold cross validation.

The baseline system performs no NL processing. For the relevance phase, this baseline system flags an entry as relevant 14% of the times, since that is the

proportion of relevant entries estimated by prior corpus analysis described in the prior section. For the selection phase, it chooses from the set of target tasks one or two tasks at random, and randomly ranks them. The baseline system does not generate any associations.

We also compared the performance of our system with simpler systems with minimal natural language processing capabilities. These systems use simple bag of words matching between the to-do entry and the task paraphrase patterns. We built two variants. One variant normalizes words from their morphological variations, using the well-known Porter stemmer [Porter 1980]. The second variant also uses synonyms from WordNet (synsets), in addition to using the stemmer.

Results for Task Relevance Identification are shown in Figure 4(a), measured with the accuracy metric. Our system has very good performance, over and above the performance of the other methods, with 87.6% accuracy after learning. After learning, out of 2100 entries our system classified correctly 1841 entries (flagged 160 as relevant that were actually relevant and did not flag 1681 that were actually not relevant) and misclassified the rest (flagged 37 that were actually not relevant and did not flag 222 that were actually relevant). The accuracy of a simple baseline system that always guessed relevance would be 18% for this dataset (a bit higher than the 14% relevance found for the corpus). A simple baseline system that always guessed irrelevance would be 82% accurate but to no good use as it would never do anything to assist the user.

Figure 4(b) shows the precision, recall, and F-measure metrics. Our system shows very high precision, significantly higher than the other methods from very early on the training. This means that when our system intervenes to make suggestions about a to-do entry, it is right to do so with very high probability. Recall is low for our system, as can be expected since its matches are much more restrictive than the other methods. We discuss how to address our overly constrained matching in the discussion section below.

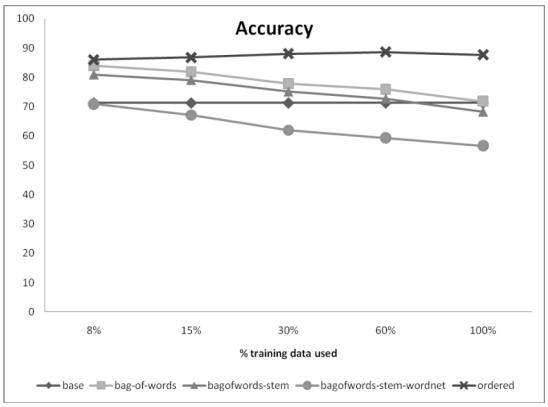
Results for Task Class Selection are shown in Figure 4(c). Our system performs best at the task, which is reflected in the t-metric results, and reaches over 90% correctness early during training. Although the g-metric shows that all methods are able to generate the right answer as part of their candidate set, the c-metric shows that our system generates a single candidate very early on during training. The user would be shown the correct target task with very high probability.

Results for Argument Association are shown in Figure 4(d). Notice that the baseline and bag of words methods do not have the ability to make such associations, so for those methods we show the amount of parameters that the correct answer would contain. This corresponds to how many edits the user would have to make. For example, "Email Bill" would correspond to sendEmail recipient="Bill", and since the to-do does not contain a topic then the user would only have to fill one argument of the target task by hand. In addition, we only reflect the edit distance of to-do entries where the selection phase for the method ranked the correct answer at the top. Our system not only does this automatically but performs extremely well, requiring between 0.2 and 0.4 user edits on average.

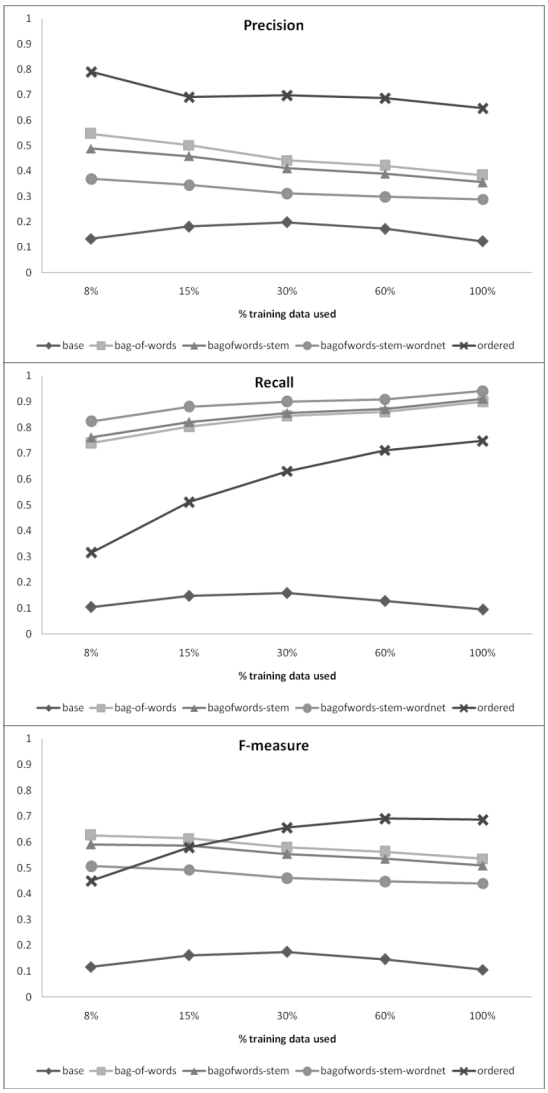
3.7 Discussion

We noted that the performance of BEAMER suffered from a limitation of our paraphrase pattern language, namely that all arguments in a paraphrase must be separated by keywords. An example is:

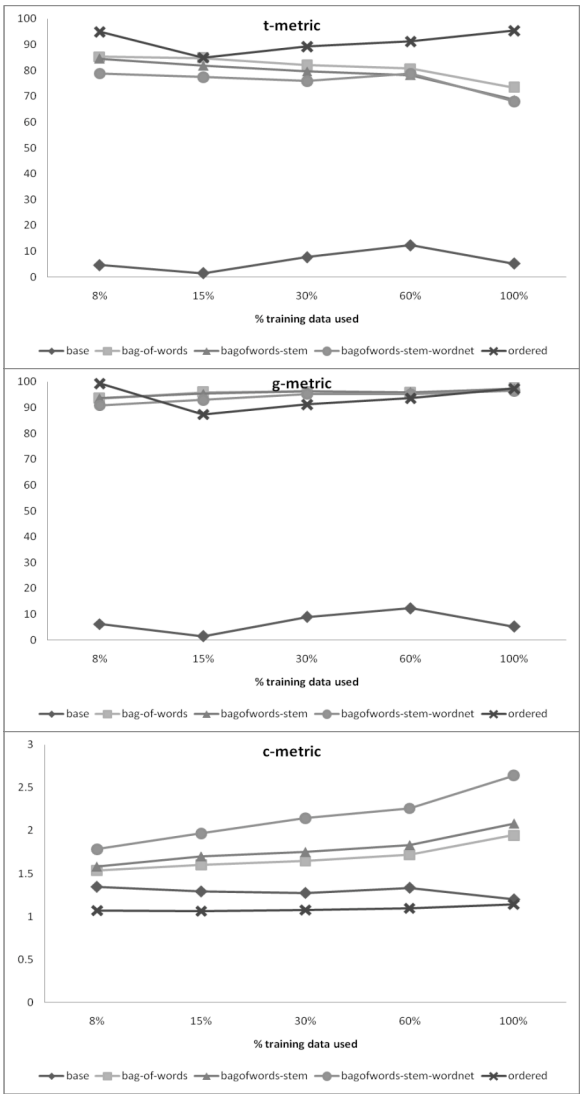
Todo Entry: Send Bill final paper
Desired pattern: Send +recipient-s +topic



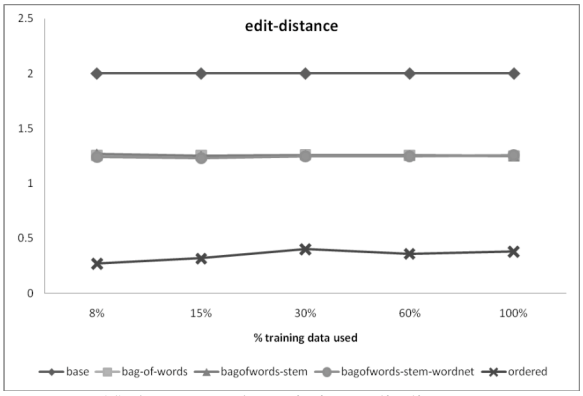
(a) Task Relevance Identification: Accuracy



(b) Task Relevance: Precision, Recall, F-Measure



(c) Task Selection metrics: t-metric, g-metric, c-metric



(d) Argument Association: edit distance

Figure 4. Evaluation of To-Do interpretation in BEAMER.

Ideally, such paraphrase pattern could be handled if the system were able to detect chunks in the to-do entry and map them to each of the arguments. In this case, the system could identify *"Bill"* and *"final paper"* as chunks and map each to the respective argument. This would also help distinguish between the two *Buy* tasks (one for equipment and another for office supplies). Off-the-shelf chunking and parsing tools are difficult to use in to-do entries because to-do entries are not complete or well-formed sentences. Adaptive parsers could be useful for this task.

An alternative would be to expand our paraphrase approach to the recognition phase. That is, to collect paraphrases for ontological objects that can give us type information. For example, we would collect paraphrases such as *"Bill"* and *"Bill H."* that refer to the ontological object WilliamHill which is of type Person. By combining recognition and matching, the system would be able to use paraphrase patterns without separating keywords.

3.8 Further Corpus Analysis of Office To-Dos: Collaboration

Many office tasks are inherently collaborative. Whether scheduling a meeting, writing a portion of a document for a colleague, or asking a family-member to pick up milk at the grocery store, tasks often require the interaction of multiple people in order to be accomplished. Indeed, a central part of task management is the tracking of how tasks have been delegated and shared.

To confirm this intuition, we performed an analysis of our to-do office corpus focusing on collaboration aspects of the tasks.

We manually checked the entire corpus and found that 17.5% of the to-dos were collaborative in nature. Either the to-do referred to a task to be accomplished for another per-son (e.g. read John's document), scheduling a meeting (e.g. discuss program with Mark), or assigning a task to another person (e.g. email Mary about John). While 17.5% is a significant percentage of the tasks, we believe that this under-states the number of collaborative tasks. From our observation, it seems that many tasks, while not specifically identifying collaborators, are sub-tasks of larger collaborative tasks. For example, background reading necessary to participate in a meeting.

Thus, given the collaborative nature of tasks, social relationships provide a key structure for tasks. However, this social structure is not explicit in common task management tools such as to-do lists and calendars. Social relationships are more evident in email through addresses and have been taken advantage to create task management interfaces [Bellotti et al. 2003]. However, email addresses do not capture the nature of the social relationships or information about the participants themselves. Groupware systems do a better job of explicitly capturing social relationships but are limited to the particular organizations that adopt them.

To address this lack of social relationship information, we decided to explore to-do management in the context of a social networking site. Before reporting on that work, the next section presents an analysis of to-dos in a personal setting in contrast with to-dos in an office environment discussed so far.

4. ELABORATING TO-DOS: EXPERIENCES WITH TO-DOS FOR PERSONAL TASKS

We wanted to explore and contrast the to-dos that we had found in a controlled office environment with the to-dos that occur in a more ubiquitous way as people carry out daily personal tasks.

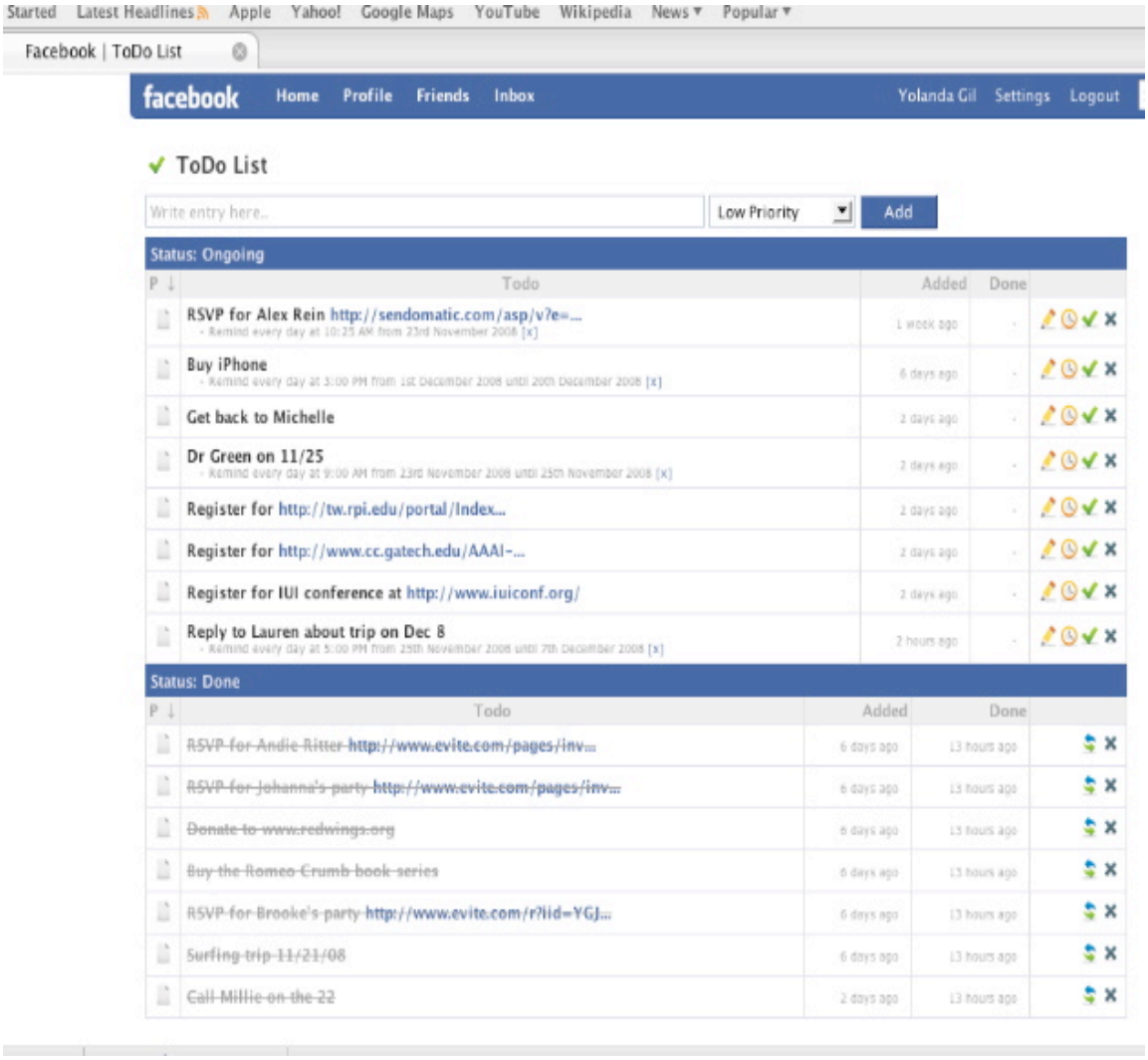


Figure 5. User Interface to collect personal to-dos.

In addition, we wanted to target the Web as a substrate for automation. For example, a system could suggest links from user to-do entries to web resources that would be helpful with their tasks. In the work described above, the amount of to-dos that we found could be automated by agents is relatively small. We suspected that we could find further opportunities for automation in the context of the Web.

Although a number of to-do management applications exist, it is hard to obtain access to such data. Therefore, we developed a to-do management application, instrumented to collect data from actual use. The application provides standard to-do features including setting dates, priorities, categories, and comments. Figure 5 shows the user interface of the application.

To ensure that the data collected would be about personal to-dos, we developed the application within a popular social networking site, since users keep a lot of personal information in it and visit it constantly so we thought it would be a convenient substrate to investigate personal task management tools. We also thought that eventually the applications for the site would themselves be able to

provide automated capabilities. We obtained preliminary feedback from a small user group before releasing it more broadly. The application has around one hundred monthly active users.

4.1 Corpus Analysis of Personal To-Dos: Opportunities for Assistance

We did an analysis based on the initial user group for this application. We analyzed 1500 to-do items collected from 325 people. Of those people, 100 were stable users of the application.

This analysis revealed a great deal about the kinds of tasks that people jot in their to-do lists, and about the potential for assistance. Some observations are:

- **Many tasks had very coarse granularity.** Examples are "Get Christmas presents", "Study nursing", and "Get back in shape". These are high-level tasks that involve many substeps and activities. There are no concrete first steps enumerated, which would be useful for a user to get started on the overall goal expressed in the to-do.
- **Many tasks that were concrete had only some aspects that could be automated.** For example, "Write Christmas cards" or "Read Dune" could involve some on-line purchasing that could be automated but most of the activity was meant to be done by the person herself.
- **Some tasks could be fully or mostly automated.** Examples include "Renew my driver's license", "Buy iPhone", "Rent Aliens movie".
- **Many tasks were not amenable to automation.** Examples include "Be a true Christian" and "Go to the mall".
- **Entries were in a variety of languages.** Even though the application had all its labels and titles in English, we found that many users adopted it to jot to-dos in their own vernacular in addition to English.
- **Many entries referred to web sites.** For example, an entry for "Buy Borges book" would include a URI for a bookseller's web site.

As we expected, the to-dos collected in the social networking site mostly focus on personal tasks that encompass leisure, study, and general activities. Work-related to-dos are extremely rare. Both corpora contain obscure entries that can be only understood within a specific context.

There were a number of major differences in how the two corpora were collected. First, the office to-dos were collected and accessible only when the user was working at their computer. In contrast, the personal to-dos were only available when users were logged into the social networking site and thus in many cases not from the office workplace. The office data was collected during regular evaluation periods, each lasting several months. There were twelve users who were pre-selected for us. In contrast, the personal to-dos were collected continuously. Users were self-selected as they came themselves to the site, and there were more than one hundred of them.

4.2 Extracting Structured Task Knowledge from Web Repositories

We wanted to explore the possibility of assisting users to elaborate to-dos, decompose them, and break them down into smaller tasks that may be more amenable to automation. Popular methodologies to manage to-dos, like GTD (Getting Things Done [Allen 2001]) or the Pomodoro technique [Cirillo 2006], suggest to make sure that items on to-do lists are "actionable," i.e. that they can be done and checked off in a reasonable timeframe. The Web offers plenty of websites with guides and manuals for tackling an enormous number of tasks, and those sites could be exploited to provide better assistance to users with their to-dos.

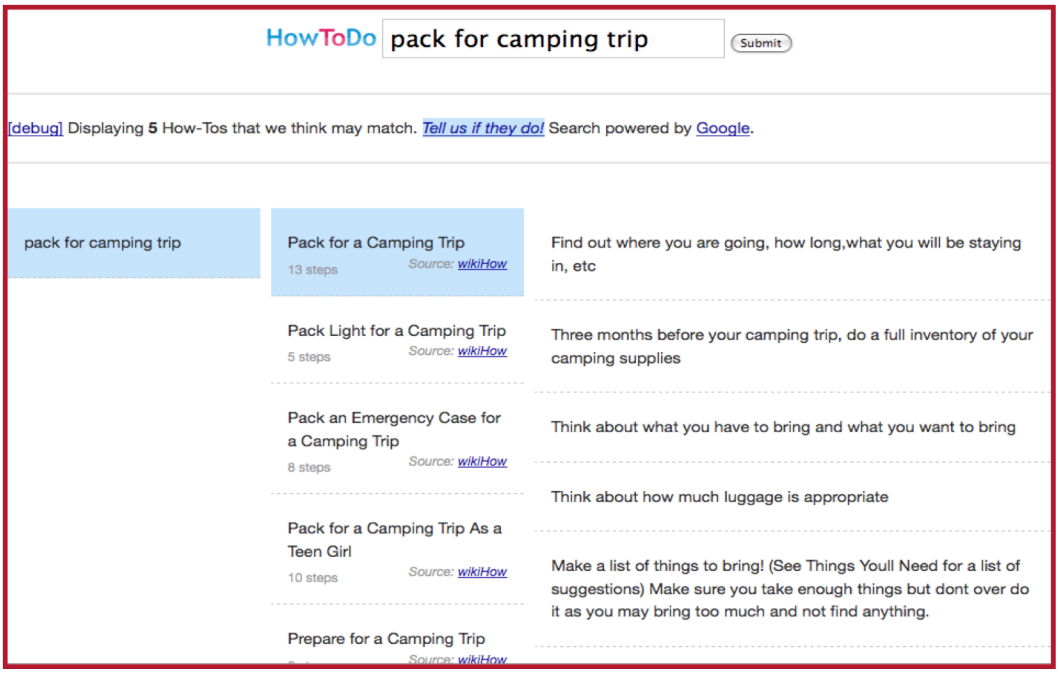


Figure 6. A screenshot of HowToDo, showing how-to data extracted from web sites. On the left are some user-entered to-do entries, the middle column shows a list of possible matches for the currently selected entry, and the right row displays the substeps for the selected match.

We implemented this type of assistance as a standalone web service, called HowToDo (http://howtodo.isi.edu). We envision that to-do applications can, once a new to-do is entered into the system, search automatically for matching how-tos from this service, present decompositions of the to-do into steps to the user, and allow the user to quickly incorporate those steps as to-dos (either in addition or instead of the original to-do). This would provide assistance by breaking down otherwise daunting tasks like "get a driver's license" down into more manageable and actually actionable tasks like "Schedule the drivers license test appointment" or "Pass a vision examination".

There are many different resources on the Web to find knowledge on how to perform specific tasks. In order to create an initial set of a relevant number of how-tos for our HowToDo repository, we have turned to one of the bigger websites directly aiming at collecting how-to knowledge, wikiHow (http://www.wikihow.com). wikiHow is a community-built site collecting how-to manuals. It is multi-lingual, even though the English part vastly outnumbers the rest. wikiHow holds currently more than 85,000 articles written collaboratively by more than 200,000 registered users under a Creative Commons license. It has more than 25 million unique visitors per month.

Articles on wikiHow are usually uniformly structured: after a short introduction we find a list of steps, which in turn are often further detailed. We use the steps to offer the more detailed subtasks. The articles end with a few general notes, links to external resources, and also a list of related wikiHow articles. Some of the steps themselves may further link to external resources or further wikiHow articles.

Table III. Matches on the evaluation samples.

	Office to-dos	Personal to-dos
True positives	16%	36%
Top ranked	4%	24%
False positives	27%	27%
None	57%	37%

Figure 6 shows a screenshot of HowToDo. HowToDo takes a to-do entry as input and returns a ranked list of possibly matching how-tos and the steps for each of these how-tos. A how-to usually has half a dozen to a dozen steps.

HowToDo has a matching component that takes as input an incoming to-do entry and retrieves and ranks fitting how-to articles from its repository. The current matching algorithm is based on a web search service. The results are then further filtered and processed. Whereas the resulting steps are often merely an extract from the original how-to article, the search uses the complete full text and the article's link structure in order to calculate the ranking. Note that the matching approach is HowToDo is different from BEAMER, as the task is quite different. BEAMER maps a to-do to agent capabilities through the paraphrases of those capabilities, and then further maps portions of the to-do into the arguments of the capabilities. HowToDo is matching a to-do with an entry for a procedure that consists of a fully textual description.

The HowToDo web service is accessible as a RESTful interface providing results in JSON (http://howtodo.isi.edu/apidoc.php). We expect a custom-made interface combining to-do and how-tos to be particularly beneficial on mobile devices, which are often used to provide ubiquitous access to a user's to-do list.

4.3 Evaluation Results for To-Do Elaboration

For this evaluation, we manually tagged a total of 480 to-dos according to what how-tos they matched. We used a sample from the office and personal to-do datasets that we collected. For the office dataset, we took a sample that was previously used for the evaluation of BEAMER in order to also compare the coverage of HowToDo. From the personal to-do dataset we collected 300 random entries (only omitting entries not in English). We manually processed every entry in the corpus and tagged them by either stating that all suggestions were bad matches, that no suggestions were made, or the rank of the first matching suggestion (sometimes, several suggestions matched).

We formulated the following hypotheses: 1) only a small, but still interesting fraction of to-do items will be matched by HowToDo, 2) the system will perform significantly better on the personal to-dos than in the office to-dos, 3) the system will rather return no how-tos at all than irrelevant how-tos, and 4) the overlap between the set of to-do items that are understood by agents and the set of to-do items that can be assisted with HowToDo is small.

An overview of the results is given in Table III. "Good" gives the percentage of true positives: it means that there was a match between the to-do entry and one of the top five ranked suggested how-tos. "Top" indicates the percentage of entries where the top ranked how-to was the correct result (thus "Good" contains "Top"). "Bad" lists the percentage of false positives: it means that the system did return results, but none of the top five ranked fitted to the to-do entry. We have been rather

strict: the system often suggested thematically related, interesting how-tos, but not one actually breaking down the given task (like matching ”Banana cupcake” with ”Make Vegan Banana Muffins”). Finally, “None” lists the percentage of entries where no how-to was suggested. As a further result, we compared the set of to-do entries tagged in the office corpus as having the potential of being automatically assisted by agents, the latter being 14% of the entries [8]. Only 3% of the sample was both amenable to automatic assistance and matched by HowToDo with a fitting how-to. This shows that the two approaches towards assisting the user with their to-do lists – automating the tasks, explored in Section 3, and matching with how-tos, as suggested in this work – are highly complementary, pointing towards a hybrid solution for optimal coverage.

Our first hypothesis was correct, and the evaluation actually exceeded our expectations for personal to-dos. This is a promising result, but we have to remember that we only evaluated the matching of the to-do entries to the how-tos, and not the usefulness of the suggested, individual steps.

The two datasets exhibit very different properties: in the office to-dos case, the number of false positives outnumbered the good suggestions, and even with the good ones, most of the time the user had to look through a number of suggestions before finding the correct one. In the personal-to-dos case, though, not only the number of matched entries was significantly higher, but also in most cases, when there was a match, the match came up as the top-ranked suggestion. This confirmed our second hypothesis: this approach works much better on the personal to-dos.

Even though the number of false positives is comparable, the false positives in the personal to-dos case were still often considered as interesting to the query, related to the topic at hand, or outright funny (matching ”world domination” to ”Win at the Game Risk” or matching ”create task to-do list app” with a work-saving ”Use Remember the Milk”). In turn, in the office to-dos case the false positives seldom had anything to do with the task, often providing an irritating distraction for the task – a consequence much less serious for users managing personal to-dos.

We assume that this is due to the particular repository of how-tos we have used. The office to-dos contain many references to project names, persons, specific forms, and procedures. Often projects are given names that are also common words – this was not the case in our corpus, thus leading to the high rate of 57% of entries not matched at all. We assume that if the projects were given names that are also common English words, we would have had a drastic increase in the percentage of false positives. But even without this further complication, our third hypothesis was shown wrong. The number of false positives was very high.

As we have seen the fourth hypothesis was shown correct on the office data: agents and HowToDo are indeed complementary in their potential. In an office setting, we could exploit the different characteristics of the to-dos in two ways. First, we could provide a more intelligent pre-processing of the query. A knowledge base containing the names of projects, people, and other relevant entities could be used to decrease the number of false positives. Second, many large corporations already have a repository of business processes or best practices. Instead of using the general interest wikiHow repository, such a system within a corporation should naturally use their own repository, this providing a natural interface to access this knowledge for users relying on a to-do application.

Another shortcoming of the current system is its lack of parameterization. Whereas wikiHow provides how-tos like ”Move” or even ”Move to a Bigger City for a New Job”, not many cities have a dedicated article on how to move there. A knowledge-based backend could expand the query by recognizing parts of the query

that could be regarded as parameters and provide with a more general match, if a specific one is not found.

4.4 Discussion

We investigated how to assist users with their to-dos by matching to-do entries with how-to articles automatically and use them to elaborate to-dos. We implemented the approach and provide the implementation as a web service, so that it can be integrated in external systems. We also provide a website, HowToDo, as a demonstrator of the service, so that the idea and implementation can be easily tested. We evaluated the implementation using two different corpora, one from office to-dos and another from personal to-dos.

Even though ”Move to Amsterdam” not only yields the helpful how-to on ”Move to the Netherlands”, but also the very practical and relevant ”Change the Timezone in Linux” and ”Talk to a Boy While Bicycling”. We found that the system performs very differently on the two corpora. For personal to-dos, the system is able to provide users with support in one out of three to-do entries, whereas for office to-dos the ratio drops to one out of six.

The current implementation is based on the repository of how-tos created by the wikiHow community. We eventually envision the web containing a huge collection of How- Tos, from the very specific – like ”deliver furniture to 3544 Jones Av, Apt 27” – to generic ones, like the ones we find today. These how-tos can be further annotated with rich metadata, helping to discover and parameterize them, and even connect them to services on the Web so that some parts of the to-dos can be resolved with a single click. HowToDo is a prototype implementation of this vision.

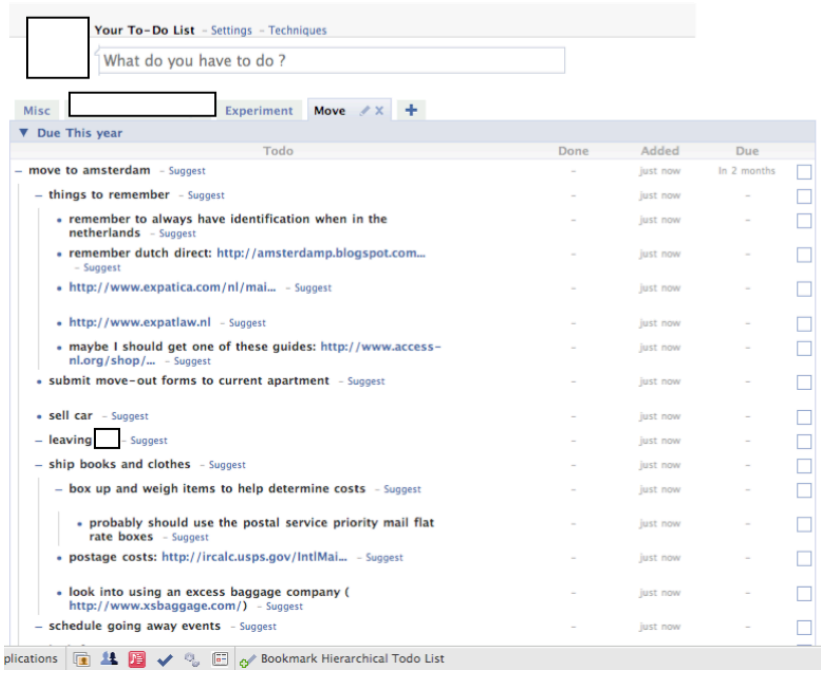
An important thing that we learned with this tool was regarding mechanisms to automate user to-dos. To automate to-dos we had planned to use “apps”, the applications developed for users of the social networking site. For example, there are apps for calendar management and for sending emails and messages. Although these apps could in principle be used to automate tasks for users, we found that the apps that we thought would be relevant for automation did not have the appropriate hooks for us to integrate with.

5. COLLABORATIVE TASKS: EXPERIENCES WITH TO-DOS IN SOCIAL NETWORS

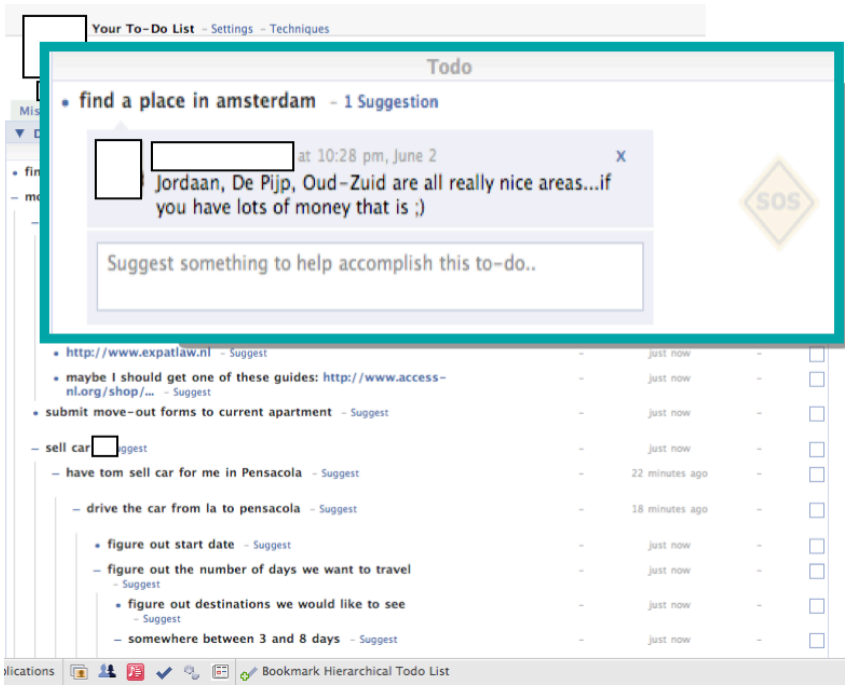
Tasks are inherently collaborative and are effectively hubs for interaction among people. Whether scheduling a meeting, writing a portion of a document for a colleague, or asking a family-member to pick up milk at the grocery store, tasks often require the interaction of multiple people in order to be accomplished. Indeed, a central part of task management is the delegation of tasks and the tracking of status when shared. Given our observations with office and personal to-dos with respect to the collaborative nature of many tasks, we embarked in an exploration for a more social and collaborative setting for to-do lists.

We developed a new version of the system that took advantage of the possibilities for sharing that are available in a social networking site (http://facebook.com). By embedding it within a social networking site, the application has complete access to an explicit social graph. Furthermore, it is in an environment that users are comfortable with and use often.

This application provides standard to-do features that our baseline application had. At the top of the application, there is an entry box for entering to-do items, modeled after a status update box. Each to-do item can be assigned a priority, category, due date and be marked as completed. Users can organize their to-do items into personal categories using tabs.

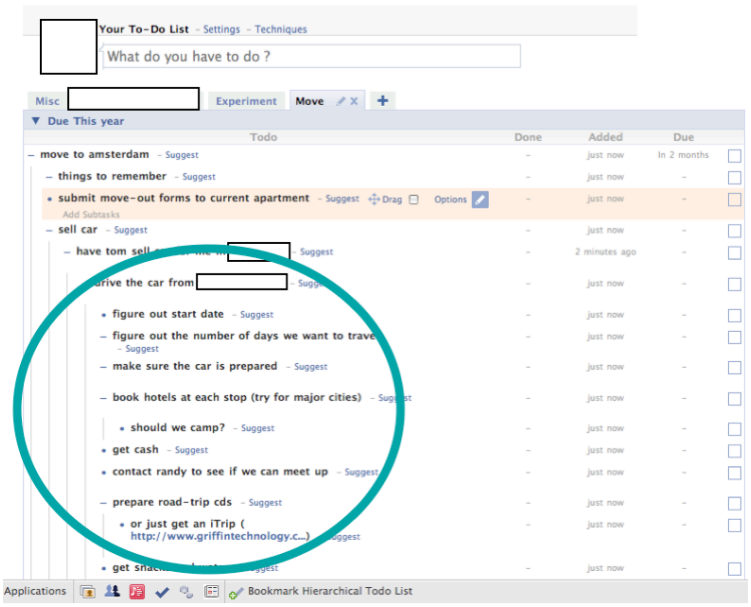


(a) Users elaborate their to-dos with subtasks, and note pointers to useful Web sites.

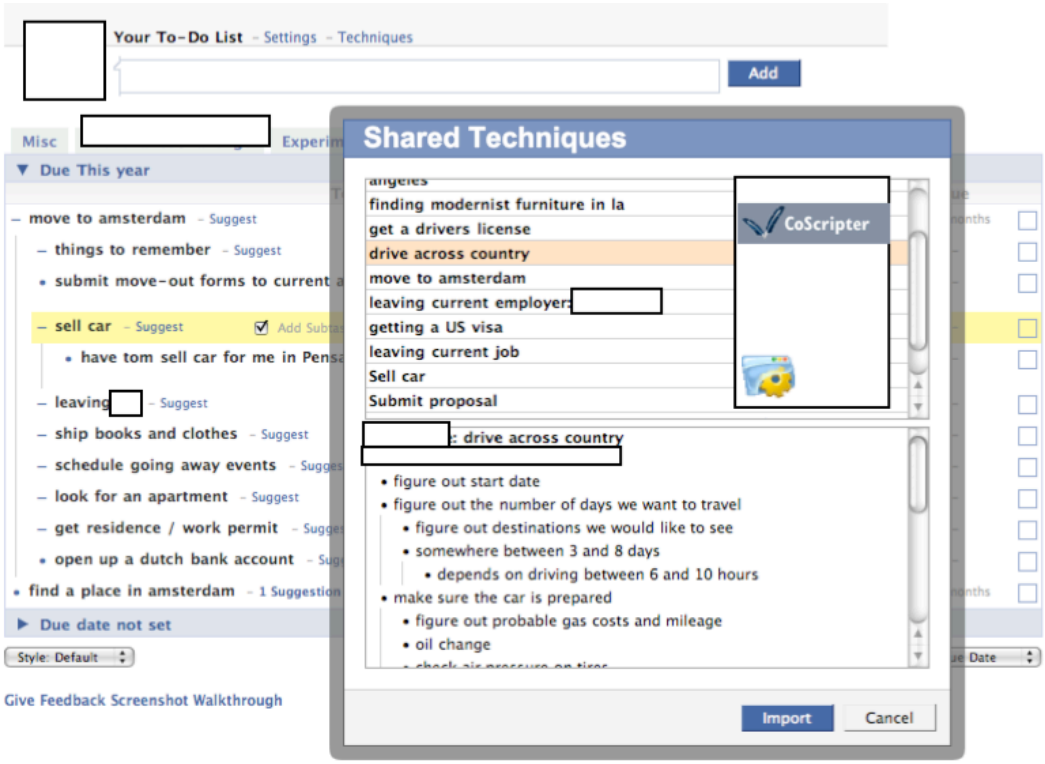


(b) Users can post their to-dos to their social network and get help from other users.

Figure 7. Users define hierarchical to-dos to decompose tasks, and can share to-dos to get help from friends in their social network.



(a) Users can share their "know-how" as techniques that are reusable by other users.



(b) Users can search a repository of techniques shared by other users.

Figure 8. To-do decomposition techniques can be shared in the social network.

Recall that we observed that finer granularity of to-dos is highly desirable, as more specific tasks are more amenable to assistance. Therefore, we allowed for hierarchical to-dos in the interface so that users can decompose tasks in terms of their constituent subtasks. The set of to-dos that are active at any given time are often part of enveloping tasks. This allows users to express tasks at coarser and finer levels of granularity, giving the task a high level coarser description when it is first jotted down and drilling down into details when the task is being pursued.

Figure 7(a) shows a snapshot of the user interface. The to-dos include pointers to web resources as URIs, as well as web scripts. In this example, some tasks must be done by the user while others are to be automated by IBM's Co-Scripter and iMacros for Mozilla's Firefox. Finding web scripts and agents that could assist the user in accomplishing their to-dos is useful but only a limited subset of to-dos could be addressed in this manner. For many to-do entries, finding other users that can assist in some way reflects the way many to-dos are accomplished in practice.

There are many opportunities for individuals within a social network to provide help with tasks, such as delegating, collaborating, and providing know-how about accomplishing tasks. Anecdotally, these sharing features have been of real use to the users. In particular, we saw users organize everything from evenings out to LAN parties.

First, we allowed sharing to-do list entries. Users can share their to-do lists with everyone on the social networking site, people just in a particular network group (i.e. larger organizations of people like a university), or with only their friends. We also maintain which users comment on other users shared to-do entries to track which users are actively sharing tasks.

Second, users can post what we term a "SOS", which is a broadcast to all the users friends that they need help with a particular task. Figure 7(b) shows this kind of assistance, where the user had shared a to-do item "Move to Amsterdam" and views the response from another user in their social network.

A third form of sharing to-dos is by sharing what can be called "personal how-tos". When a To-Do is accomplished, users can share the process they followed with others (think of this as sharing procedures just like you would share pictures). To enable users to share their to-do lists with other users in a reusable fashion, we introduced the notion of techniques. A technique is a parameterized hierarchical to-do list that is publicly available. A user can export from the application a portion of their to-do list as a technique. Likewise, users can import a technique into their own to-do list. Figure 8 shows the user interface where techniques are created (a) and searched (b). In the figure, the technique "drive across country" is selected. Note, at the top of the technique an argument (friends and relatives) is listed. When exporting the technique, the user selected a set of words that were anonymized as parameters/arguments. When another user imports the technique, they will be asked to fill in the argument. Furthermore, the system keeps track of the provenance of to-dos created from importing techniques. When creating a technique, users are notified that it will be completely public and available generally on the Web.

5.1 Social Task Networks

We define Social Task Networks (STNs) as tasks that are created and shared across users in a social network. The name also reflects that the structure we have given the to-do list is as a decomposition much in the hierarchical task networks (HTN) style of AI planning. Note that it is important that to-dos are decomposed into smaller tasks. This allows users to express to-dos in enough detail to allow tracking and sharing in the network.

STNs capture *task networks* through the following mechanisms:

- networked subtasks through links from tasks to subtasks in a hierarchy
- networked web tasks through links from tasks to web resources expressed as URIs
- networks of user tasks through links from a user to their personal tasks

STNs capture *social tasks* through the following mechanisms:

- networks of user tasks to other users through links from a user's shared task to other users that volunteer to provide assistance
- networks of user tasks to other user tasks through links between shared and reused techniques

STNs reflect the role of users personal tasks in their social fabric, and capture the role of web resources in the accomplishment of tasks.

5.2 Experimenting with Social Task Networks

To gain an initial insight into how a social network of users might interact with the STN paradigm, we asked two groups of users to make use of this version of the application. The first group of users (Group A) consisted of twenty-eight users and they were asked to perform the same task while using the application. All users had prior experience with the social networking site. We asked the user to perform the following activity for three days to start and then use the application further if it was useful to them:

> *"Imagine that you are planning a move to Los Angeles. Use the application, to write a detailed to-do list of the things you would need to get done to achieve this goal."*

We gave a brief guide to these users highlighting the various features of the application. Since this was an artificial task, we followed up with reminder emails to all the users encouraging participation over the course of the initial three-day period. The second group of users (Group B) consisted of five new students and the project assistant involved introducing them to our institute. We believed that because of their many shared and similar tasks, they would be good candidates for using the application. In particular, we thought this might allow the project assistant to expend less resources in answering the same questions repeatedly.

These groups of users provided useful insights into how people might prefer to manage their tasks. First, we learned that people are very concerned about the privacy of their to-dos even in their social network. They prefer to be very selective regarding who sees their to-dos. Perhaps this kind of information is more personal than other things that are shared in social networking sites. Second, we found that users were more inclined to resort to face-to-face interactions rather than using the system. This might be due to the proximity of the users in physical space, being in the same office building. Finally, it is clear that without having significant help from the system the users will not be inclined to jot a lot of to-dos or to be very specific about them. As we mentioned above, the more the system will help them the more we expect users will be enticed to jot to-dos in a more specific and shareable manner.

Figure 9 depicts the Social Task Network of the user experiment just described. It shows users, their to-dos, and web resources, and the links among them. Each top-level to-do item (i.e. one without a parent) is connected to the user that created it. Users are connected to their friends by red lines representing the social network. Web resources (URIs) are connected to each other through the social and task networks.

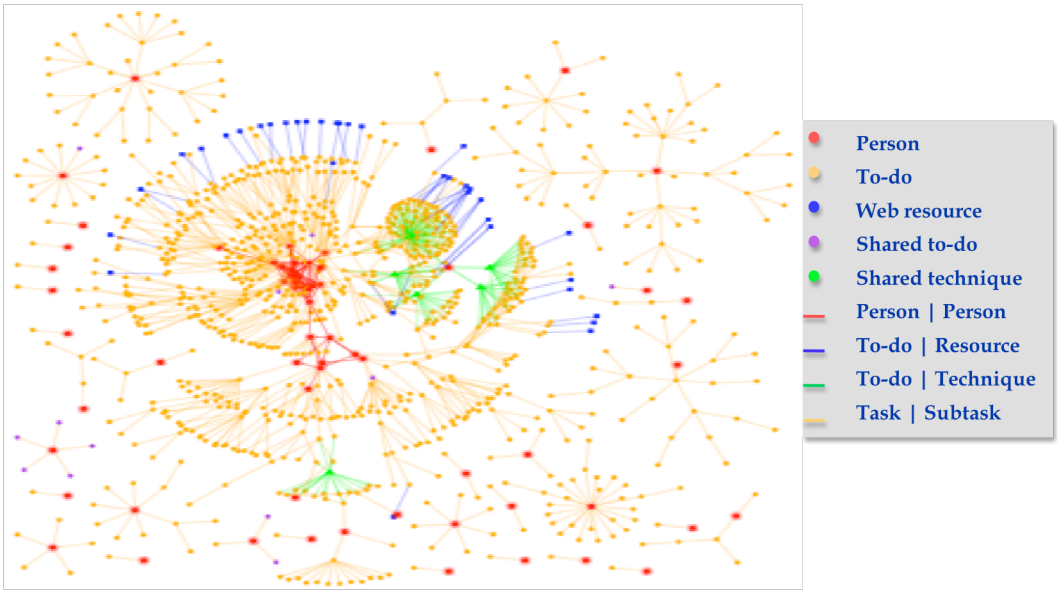


Figure 9. A visualization of a Social Task Network, showing links from people to the tasks of others in their social network.

5.3 Task Knowledge on the Semantic Web

As discussed above, tasks can provide an important alternative mechanism for linking web resources and thus add additionally contextual knowledge to the usage of resources. For example, knowledge about businesses can be bridged by the notion that these locations are involved in the execution of the same task. To make this information on the Semantic Web, we exposed part of the information gathered from users in an Open Task Repository. In particular, because of their public nature, we chose to expose techniques in this repository.

The repository is implemented using Triplify [Auer et al 2009] a relational database to RDF mapping system. To integrate techniques with the rest of the Semantic Web, we reuse existing ontologies. For describing tasks, the Semantic Desktop Task Model Ontology [Bernardi et al 2007] created by the NEPOMUK project [Groza et al 2007] was used. Through the integration with this ontology, we aim to allow the Open Task Repository to integrate with existing Semantic Desktop personal information management tools [Sauermann et al 2008].

In addition to reusing ontologies, the repository follows Linked Data principles [Bizer et al 2009] so that techniques and their constituent tasks can be easily referenced, retrieved, and processed by external applications. Following the fourth rule of Linked Data (``Include links to other URIs. so that they can discover more things."), we extended Triplify to perform an extraction of all links within a task description. This is expressed using the Dublin Core references property [6]. Hence, task descriptions and the techniques they belong to are explicitly connected to the Web. The Open Task Repository can be browsed using generic Semantic Web browsers such as Tabulator [Berners-Lee et al 2008], as shown in Figure 10.

This Open Task Repository can be seen as a bridge between popular but closed social networks and decentralized open social networks.

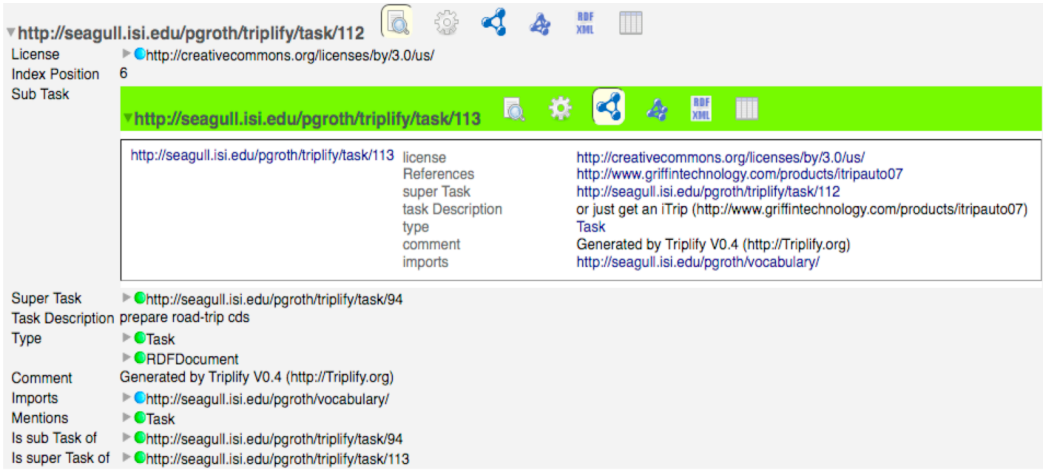


Figure 10. A View of the Open Task Repository using Tabulator.

5.4 Opportunities for Assistance in Social Task Networks

There are many opportunities for intelligent assistance with to-dos in social task networks. The Open Task Repository itself contains a wealth of data that could be exploited to make suggestions to any given user. A user could be shown the most popular methods to accomplish a particular kind of to-do, how long it took to accomplish them, and how often it was done with help from other people.

Perhaps the most interesting kinds of assistance would result from exploiting the social network itself. For example, each user's to-do could be matched against to-dos of the user's social connections. This would enable the system to make suggestions, from simple things such as informing the user that a particular friend has recently had that same to-do, showing the user the particular decomposition that their friend used, and adapting that task decomposition based on the particulars of the user's to-do. The system could also look at the way the user has handled similar to-dos in the past, suggesting sharing options accordingly and noting who had helped them with those past to-dos.

6. CONCLUSIONS

We have illustrated the use of several alternative sources of knowledge and assistance for tasks that users manage in their to-do lists. Our research has explored how users can be assisted with their to-dos by automated procedures (agents or shared web scripts), by textual instructions on how to decompose a task, or by other users in a social network.

We showed an approach for to-do list interpretation by matching to-do entries to the capabilities of available agents using a paraphrasing approach. We did this work on a corpus that we collected of to-dos in an office assistant system. Our analysis revealed that 14% of user to-dos could be matched to agent capabilities in the system, and that our approach achieved an accuracy of 88% when matching to-dos. We also showed a complementary type of assistance where to-do entries are matched with repositories of how-to instructions extracted from the Web. We compared the relevance of this type of assistance to the office corpus, as well as a corpus of personal to-dos that we collected. Finally, we explored collaboration issues of to-dos embedded in the social networks of users. We created a to-do application that allows users to share to-do entries and to-do lists within their social network, in order to gather recommendations and suggestions for doing the tasks, and discuss

them with friends. We show the emergence of a Social Task Network, a graph of users, the to-do items they have created, sub-steps of the to-dos, external Web resources, and how they are all connected.

We envision that a comprehensive, intelligent to-do assistant will implement all these and other types of assistance in order to cover as many different to-do items as possible in support of the user. We discussed in this paper a range of broader challenges in interpreting textual to-dos, using context and common knowledge about user tasks, user interaction design, and coordination of to-dos with other activities and users.

To-do list assistance could greatly benefit from a larger body of work in user interfaces on utility estimation of user interaction and various approaches to adjustable autonomy [Horwitz et al 1999; Maheswaran et al 2004]. There are interesting opportunities for learning about appropriate user interactions, about the types of assistance that are always declined or that certain users have a much higher threshold for accepting assistance.

Important design criteria for to-do lists are suggested by user studies in cognitive psychology and human factors. It would be useful to integrate the to-do list functionality with calendar and email systems and automatically extract from them to-do list entries as well as notice completed ones as the user informs others or manipulates their calendar. Automatic spotting and interpretation of user task statements and to-do entries can play an important role in the future of better integrated, more helpful office assistants.

ACKNOWLEDGMENTS

We gratefully acknowledge funding for this work by the Defense Advanced Research Projects Agency (DARPA) under contracts HR0011-07-C-0060 and NBCHD030010, and by the National Science Foundation (NSF) under grant number IIS-0948429.

We thank Karen Myers and Ken Conley for their feedback on this work and its integration with the CALO architecture. We also thank Michael Freed and Dustin Moskovitz for their useful feedback on this work.

REFERENCES

ALLEN, D. 2001. Getting Things Done: The Art of Stress-Free Productivity. Penguin.

ALLEN, J.F. SCHUBERT, L.K.; FERGUSON, G.; HEEMAN, P.; HWANG, C.H.; KATO, T.; LIGHT, M.; MARTIN, N.G.; MILLER, B.W.; POESIO, M.; AND D.R. TRAUM. 1995. The TRAINS Project: A Case Study in Defining a Conversational Planning Agent, Journal of Experimental and Theoretical Artificial Intelligence.

ALLEN, J.F., B. MILLER, E. RINGGER AND T. SIKORSKI. 1996. A Robust System for Natural Spoken Dialogue, Proc. 34th *Assoc. for Computational Linguistics (ACL).*

ALLEN, J.; FERGUSON, G. AND A. STENT. 2001. An architecture for more realistic conversational systems. in *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1-8, Santa Fe, NM, January 14-17, 2001.

AUER, S.; DIETZOLD, S.; LEHMANN, J.; HELLMANN, S. AND D. AUMUELLER. 2009. Triplify - light-weight linked data publication from relational databases. In 18th International World Wide Web Conference (WWW).

BARZILAY, R., AND L. LEE. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT).

BELLOTTI, V.; DUCHENEAUT, N.; HOWARD, M., SMITH, I. 2003. Taking email to task: the design and evaluation of a task management centered email tool. ACM Conference on Human Factors in Computing Systems (CHI).

BELLOTTI, V., B. DALAL, N. GOOD, P. FLYNN, D. BOBROW, N. DUCHENEAUT. 2004. 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).

BERNARDI, A.; BRUNZEL, M.; GREBNER, O.; ONG, E.; RISS, U. AND T. ROTH-BERGHOFER. 2007. Task Management Model - Deliverable 3.1. Technical report, Project Nepomuk. Available from http://nepomuk.semanticdesktop.org/xwiki/bin/download/Main1/D3- 1s/D3.1 v10 NEPOMUK Task Management Model.pdf.

BERNERS-LEE, T.; HENDLER, J. AND O. LASSILA. 2001. The Semantic Web. Scientific American.

BERNERS-LEE, T.; HOLLENBACH, J.; LU, K.; PRESBREY, J.; PRUD'HOMMEAUX, E. AND MC SCHRAEFEL. 2008. Tabulator Redux: Browsing and Writing Linked Data. In WWW 2008 Workshop: Linked Data on the Web (LDOW2008).

BERRY, P.; CONLEY, K.; GERVASIO, M.; PEINTNER, B.; URIBE, T.; AND YORKE-SMITH, N. 2006. Deploying a Personalized Time Management Agent. *Proceedings of AAMAS'06* Industrial Track, Hakodate, Japan.

BIZER, C., HEATH, T. AND BERNERS-LEE, T. 2009. Linked Data - The Story So Far. Int. J. Semantic Web Inf. Syst., 5 (3). pp. 1-22.

BLYTHE, J. 2005. Task Learning by Instruction in Tailor. Proc. of Intelligent User Interfaces (IUI).

BRANTS, T. AND A. FRANZ. (2006). Web 1T 5-gram Version 1. Linguistic Data Consortium, Philadelphia, PA.

CHALUPSKY, H.; GIL, Y.; KNOBLOCK, C. A.; LERMAN, K.; OH, J.; PYNADATH, D.; RUSS, T. A. AND TAMBE, M. 2002. Electric Elves: Agent Technology for Supporting Human Organizations. AI Magazine, 23(2).

CHEYER, A. AND PARK, J. AND GIULI, R. 2005. IRIS: Integrate. Relate. Infer. Share. *1st Workshop on The Semantic Desktop. 4th Intl Semantic Web Conference.*

CHKLOVSKI, T. AND GIL, Y. 2005. An Analysis of Knowledge Collected from Volunteer Contributors. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI).*

CHKLOVSKI, T. 2005. Collecting Paraphrase Corpora from Volunteer Contributors. In *Proceedings of the Third International Conference on Knowledge Capture (K-CAP).*

CHKLOVSKI, T. AND P. PANTEL. 2004. VerbOcean: Mining the Web for Fine-Grained Semantic Verb Relations. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP).*

CHKLOVSKI, T. AND P. PANTEL. 2005. Global Path-based Refinement of Noisy Graphs Applied to Verb Semantics. In *Proceedings of The Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, South Korea.

CHKLOVSKI, T. 2003. LEARNER: A System for Acquiring Commonsense Knowledge by Analogy. *Proceedings of Second International Conference on Knowledge Capture (K-CAP).*

CHUNG, G.; SENEFF, S. AND C. WANG, 2005. Automatic Induction of Language Model Data for a Spoken Dialogue System. *Proc. SIGDIAL.*

CIRILLO, F. 2006. The Pomodoro Technique. Lulu.com.

CONLEY, K. AND J. CARPENTER. 2007. TOWEL: Towards an intelligent to-do list. In AAAI 2007 Spring Symp. on Interaction Challenges for Intelligent Assistants.

DUBLIN CORE METADATA INITIATIVE. 2009. Dublin Core Metadata Terms, May 2009. Available from http://dublincore.org/documents/dcmi-terms/.

FAULRING, A.; MYERS, B.; MOHNKERN, K.; SCHMERL, B.; STEINFELD, A.; ZIMMERMAN, J.; SMAILAGIC, A.; HANSEN, J. AND D. SIEWIOREK. 2010. Agent-assisted task management that reduces email overload. IUI.

FILLMORE, C.J. 1969. The Case for Case. In Bach and Harms (Eds.), *Universals in Linguistic Theory*.

GAVALDA M. AND A. WAIBEL. 1998. Growing semantic grammars. In Proceedings of the COLING/ACL, Montreal, Canada.

GIL Y. AND V. RATNAKAR. 2008. Towards intelligent assistance for to-do lists. Proceedings of the ACM Conference on Intelligent User Interfaces (IUI).

GIL Y. AND V. RATNAKAR. 2008. Automating To-Do lists for users. Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI).

GIL, Y.; GROTH, P. AND V. RATNAKAR. 2009. Leveraging Social Networking Sites to Acquire Rich Task Structure. In Proceedings of the IJCAI Workshop on User-Contributed Knowledge and Artificial Intelligence: An Evolving Synergy (WikiAI).

GIL, Y.; GROTH, P. AND V. RATNAKAR. 2010. Social task networks. AAAI Fall Symposium Series on Proactive Assistant Agents, Arlington, VA.

GLASS, J.; WEINSTEIN, E.; CYPHERS, S.; POLIFRONI, J.; CHUNG, G. AND M. NAKANO. 2004. A Framework for Developing Conversational User Interfaces. Proc. *CADUI*, Funchal, Portugal.

GROZA, T.; HANDSCHUH, S.; MOELLER, K.; GRIMNES, G.; SAUERMANN, L.; MINACK, E.; MESNAGE, C.; JAZAYERI, M.; REIF, G. AND R. GUDJONSDOTTIR. 2007. The NEPOMUK project - on the way to the social semantic desktop. In T. Pellegrini and S. Schaffert, editors, Proceedings of I-Semantics' 07, pages pp. 201–211. JUCS.

HAYES, G., PIERCE, J. S., AND ABOWD, G. D. 2003. Practices for Capturing Short Important Thoughts. *ACM Conference on Human Factors in Computing Systems (CHI).*

HORVITZ, E., A. JACOBS, D. HOVEL. 1999. Attention-Sensitive Alerting, *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence* (UAI).

JONES, S. R. AND P. J. THOMAS. 1997. Empirical assessment of individuals' 'personal information management systems'. *Behaviour & Information Technology* 16(3): 158-160.

KIRSH, D. 2001. The Context of Work. *Human Computer Interaction*, Vol 16.

KUSHMERICK, N. AND T. A. LAU. 2005. Automated email activity management: an unsupervised learning approach. Proceedings of the ACM Conference on Intelligent User Interfaces (IUI).

LIEBERMAN, H.; LIU, H.; SINGH, P.; AND B. BARRY. 2004. Beating some common sense into interactive applications. AI Magazine.

MAHESWARAN, R.; TAMBE, M.; VARAKANTHAM, P.; AND MYERS, K. 2004. Adjustable autonomy challenges in personal assistant agents: A position paper. In *Agents and Computational Autonomy*, Lecture Notes in Computer Science, 1994. 187–194.

MILLER, G. A. 1990. WordNet: An on-line lexical database. International Journal of Lexicography 3, 4 (Winter 1990), 235-312.

MITCHELL, T., WANG, S., HUANG, Y., AND CHEYER, A. 2006. Extracting Knowledge about Users' Activities from Raw Workstation Contents. *The Twenty-First National Conference on Artificial Intelligence (AAAI).*

MORLEY, D., AND MYERS, K. 2004. The SPARK agent framework. In *Proc. of AAMAS'04*, 714–721.

MYERS, K., P. BERRY, J. BLYTHE, K. CONLEY, M. GERVASIO, D. MCGUINNESS, D. MORLEY, A. PFEFFER, M. POLLACK, M. TAMBE. 2007. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine*, 28(2).

NORMAN, D. A. 1991. Cognitive artifacts. In J. Carroll, editor, Designing Interaction: Psychology at the Human-Computer Interface, pages 17--38. Cambridge University Press, New York, NY.

PORTER, M.F. 1980, An algorithm for suffix stripping, Program, 14(3) pp 130–137.

RINGLE, M.D. AND R. HALSTEAD-NUSSLOCH. 1989. Shaping User Input: A Strategy for Natural Language Dialogue Design. Interacting with Computers 1(3).

RUDNICKY, A., THAYER, E., CONSTANTINIDES, P., TCHOU, C., SHERN, R., LENZO, K., XU W., OH, A. 1999. Creating natural dialogs in the Carnegie Mellon Communicator system. *Proceedings of Eurospeech*.

SAUERMANN, L.; GRIMNES, G. AND T. ROTH-BERGHOFER. 2008. The semantic desktop as a foundation for PIM research. In J. Teevan and W. Jones, editors, Proceedings of the Personal Information Management Workshop, CHI-2008.

SENEFF, S. 1992. TINA: A natural language system for spoken language applications, *Computational Linguistics*, 18(1).

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.

SINGH, P., T. LIN, E. MUELLER, G. LIM, T. PERKINS, AND W.L. ZHU. 2002. Open Mind Common Sense: Knowledge acquisition from the general public. *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems.* Lecture Notes in Computer Science. Heidelberg: Springer-Verlag.

SMITH, D. AND H. LIEBERMAN. 2010. The Why UI: Using Goal Networks to Improve User Interfaces. Proceeding of the 14th international Conference on Intelligent User Interfaces(IUI).

VRANDEČIĆ, D.; GIL, Y.; AND RATNAKAR, V. 2011. Want World Domination? Play Risk! Matching To-Do items with How-Tos from the Web. Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI).

WANG, Y. AND A. ACERO. 2001. Grammar Learning for Spoken Language Understanding. Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding. Madonna di Campiglio, Italy.

WANG, Y. AND A. ACERO. 2002. Evaluation of Spoken Language Grammar Learning in the ATIS Domain. Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing. Orlando, Florida.

WHITTAKER, S.; BELLOTTI, V. AND J. GWIZDKA. 2006. Email in personal information management. Communications of the ACM, 49(1):68–73.

ZOLTAN-FORD, E. 1991. How to Get People to Say and Type What Computers Can Understand. International Journal of Man-Machine Studies 34(4).

Received May 2011; revised XX 2011; accepted XX 2011