Enhancing Interaction with To-Do Lists Using Artificial Assistants

Yolanda Gil, Timothy Chklovski

USC Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{gil, timc}@isi.edu

Abstract

Assisting users with To Do lists presents new challenges for intelligent user interfaces. This paper presents our approach and an implemented system, BEAM, to process To Do list entries and map them to tasks that can be automated for the user. The system then monitors the progress of the execution of those tasks and processes their completion. We use a novel approach to interpreting natural language tasks that exploits semi-formal knowledge repositories collected from web volunteers, which are broad-coverage and continuously increase in size at zero cost. An important aspect of our research is that it has been heavily influenced by an office assistant architecture that learns continuously to assist the user with new tasks.

Introduction

To Do lists are ubiquitous, whether in our handheld organizers, desktops, or in the traditional form of lists on paper 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]. 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 clearly views these as onerous additional tasks that do not help the user with their original task entered into the list.

We see an immense and largely unexplored opportunity for intelligent assistance in automatically interpreting, managing, automating, and in general assisting users with their To Do lists. This opportunity presents important challenges to intelligent user interface research. Interpreting To Do lists requires natural language (NL) processing and interpretation that, as we argue in this paper, requires new research. To Do lists also require developing intelligent assistants that can exploit those interpretations to act, anticipate, and learn to expand their skills over time. 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 latter 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].

Our initial work on providing intelligent assistance for To Do list management is part of a larger project to develop intelligent assistants for office-related tasks (www.sri.com/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 2006] as well as a calendar management system [Berry et al 2005] 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, Dietterich & Herlocker, 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. Our work to date has been 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, BEAM, has only an initial set of the To Do list management capabilities but it is fully integrated with an end-to-end execution and monitoring system and will be extended in the coming year with additional functionality. While we do not present experimental results in this paper, the overall system is undergoing a formal evaluation with several dozen users during September and October of 2006.

In this paper, we outline a research agenda for this new area, discuss the challenges that we decided to tackle, the approach we took, the architecture and components of the system, and the system's integration within the larger architecture. We end by discussing forthcoming prospects and planned research.

Intelligent Assistance for To Do Lists

We are interested in developing an intelligent office assistant that can assist users with their tasks using To Do lists as primary method of communication. The system should be proactive in identifying assistance opportunities and anticipating possible tasks which the user may not have mentioned. Various forms of assistance include:

- Act on the To Do list. This entails mapping entries to the automated tasks which 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.
- Augment the To Do list. Given an entry, the system could suggest additional preparatory tasks and sub-tasks which 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 which 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.
- Coordinate To Do lists across users in an organization. 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.
- Assist users with organization 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.

In our work, we concentrate on the first two: acting on To Do list entries to carry them out automatically, and monitoring their execution in order to alert the user about their ongoing status and successful completion.

New Challenges

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]. 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]. In our work, we build on the semantic grammar approaches in automatic interpretation of user utterances, as well as on the approach which collects paraphrases from users to extend the set of statements which 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 is 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. In our case, the set of tasks is ever expanding. If the system is extended (or learns on its own) to purchase books through the Internet, how can it identify occurrences of book purchasing 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.

Approach

Our approach is to exploit semi-structured repositories of common knowledge that have broad coverage and are continuously extended at zero cost. 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. 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. In prior work, 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].

Figure 1 shows an overview of the overall functionality of BEAM and the knowledge sources it uses. We indicate with regular lines the portions of the system that have been developed and integrated, with dotted lines the knowledge sources that have been developed but are not yet exploited in BEAM's capabilities, and with dashed lines the capabilities that are not yet implemented but which we anticipate we will be able to develop within this framework. We note that some of the components shown have been developed by others, namely the ontologies and facts, the task catalog, the execution and monitoring system, the calendar scheduling system, the instrumented desktop, and task and concept learning components [Myers et al 2006; Cheyer et al 2005; Berry et al 2005; Shen, Dietterich & Herlocker, 2005; Blythe 2005]. We also note that there are many other components being developed and integrated within the overall project that are not directly relevant to this work and are not mentioned here (more can be found at www.sri.com/calo).

The knowledge sources for BEAM include semi-formal repositories of broad knowledge that we created, as well as formal ontologies and task catalogs that were engineered and are directly relevant to the office tasks. The type, collection, and structure of the semi-formal repositories was guided by the kinds of To Do list processing that we currently do or that we envision doing in the near future. The 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". The Substeps

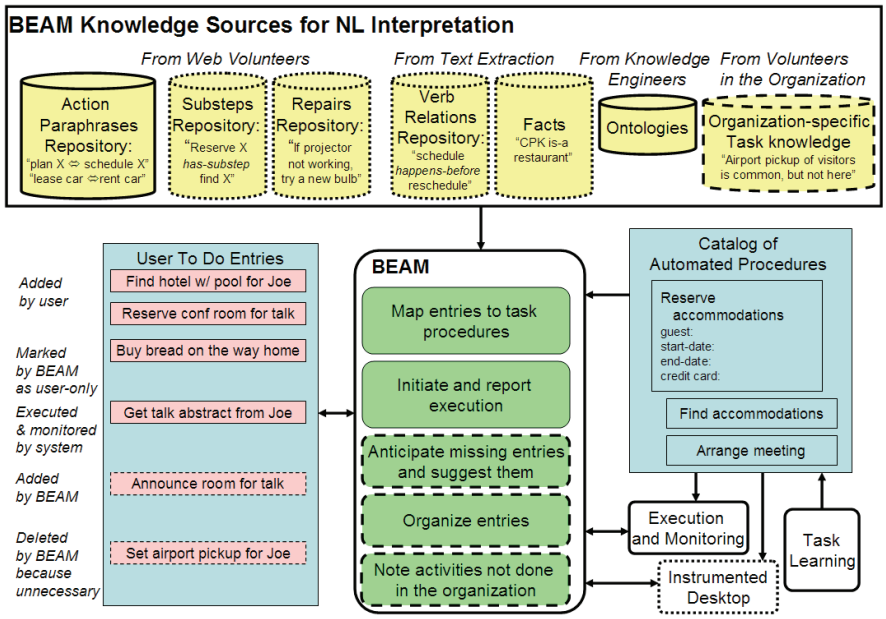

Figure 1. Overview of the overall functionality of BEAM and the knowledge sources that it uses. Dotted lines indicate knowledge sources already developed but not yet used by BEAM. Dashed lines indicate capabilities not yet developed.

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]. 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, or to action on office objects or their parts. The Verb Relations repository, VerbOcean [Chklovski & Pantel 2004, 2005], 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.

BEAM also exploits ontologies and factual knowledge bases that represent formally all the terms that the system understands. The tasks that the system knows how to automate are described in a Procedures Catalog, which states the action type, the argument slots required for each procedure, and constraints on types of the acceptable slot values. For instance, a hotel reservation task may have as slots the guest name the start and end dates of the stay.

To Do entries may not be actionable as they are stated, and BEAM has the challenge of mapping entries with incomplete arguments. For instance, arranging a lunch may require specifying for how many people; announcing a talk by a visitor may require the talk title and date. In these cases, further interaction with the user is required to elicit the additional parameters before action can be taken.

User Interface for To Do Lists

Figure 2 shows some examples of To Do list entries in the BEAM user interface. Note that, as we mentioned earlier, we make the assumption that To Do list entries have a verb header followed by arguments and constraints.

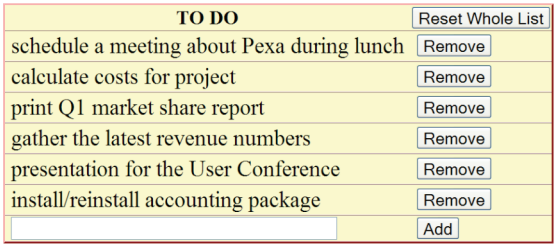

Figure 2. Screenshot of BEAM's To Do List

The To Do list items shown include: an entry that can be fully interpreted by the system, "schedule a meeting about Pexa during lunch," or "calculate costs for project," entries that can only be partially interpreted by the system, "print Q1 market share report," "gather the revenue numbers," and entries which cannot be interpreted by the system, "presentation for the User Conference," and "install/reinstall accounting package."

The first two entries, "schedule a meeting about Pexa during lunch," and "calculate costs for project" can be fully interpreted because they use vocabulary covered by BEAM and they conform to BEAM's syntactic expectations.

The entries "print Q1 market share report" and "gather the revenue numbers" can only be partially interpreted: while BEAM interprets the action stated – print a document and gather information, BEAM is not able to process in any depth the entities that are the arguments of these actions – what needs to be printed or gathered. In the case of "gather the latest revenue numbers," BEAM's syntactic analysis does yield that latest is an adjective modifying "revenue numbers", but no interpretation (mapping to a semantic concept) is found for "revenue numbers".

The final entries cannot be interpreted because they do not conform to BEAM's expectations about the input: "presentation for the User Conference" is not in the form of a verb with additional arguments, while "install/reinstall accounting package" refers to multiple actions, and mentions "accounting package" which has no interpretation.

Figure 3 shows an option of the BEAM interface which, when activated, allows users to see the portions of each To Do list entry that the system can interpret. This is a useful feature, as studies have shown that humans adapt to the vocabulary of another human or that of a system even if it is different from their own. Our expectation is that this feature will train users to stay away from using the terms and grammatical structures which the system cannot interpret. With this option enabled, BEAM's processing into the action and the arguments is displayed – eg, the first entry is broken up into the action (schedule) and three arguments. Also, all identified action and entity expressions are bolded (eg, meeting, Pexa, lunch), while other lexical items are not (eg., about, during, latest). The bolded entities are further color-coded to indicate the interpreted actions (schedule, calculate, print and gather), and entities (meeting, Pexa, lunch, costs and project).

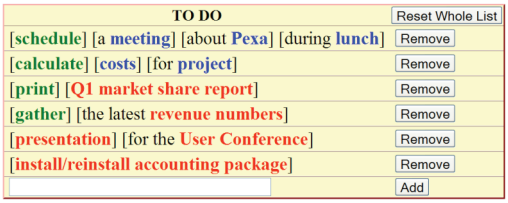

Figure 3. Screenshot of BEAM's To Do List Showing the User Recognized and Unrecognized Text Fragments

Figures 4 through 7 shows several aspects of the user interface that was developed to integrate BEAM with the execution and monitoring system (the system is described in detail in [Myers et al 2006]). Figure 4a shows the To Do list with several items. The last two items have been completed and are shown with checkmarks and in italics. The actions for which automated assistance was invoked are marked with a "C" icon (for CALO, the automated cognitive assistant). The third item, "plan conference travel" has been mapped to an existing capability of the assistant, and is being carried out. The To Do list monitors and indicates its status.

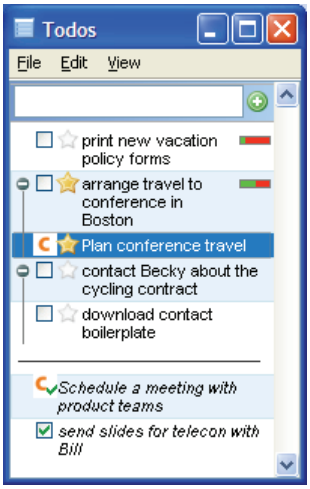

Figure 4. A To Do interface integrating BEAM and providing task interpretation and monitoring.

Carrying out the action may require a number of steps by the execution and monitoring system. The user can interact with the assistant in the execution and monitoring system, eg, to query it about the assistant is currently working on. This is illustrated in Figure 5.

Figure 5. Execution and monitoring system interacting with the user about progress on task.

In the case of planning conference travel, carrying out the action requires additional user interaction, including the entry of missing arguments about conference deadline, etc. The execution system can initiate additional interactions with the user; a form-based interaction about the conference details is illustrated in Figure 6.

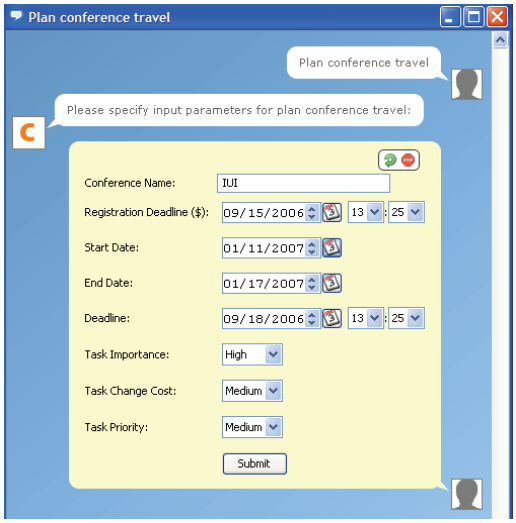

Figure 6. As part of its interaction, additional parameters can be elicited from the user in the process of running the execution and monitoring procedure.

Finally, once the task is successfully completed, the user can monitor this in the To Do interface. The automated task is displayed as completed, as shown in Figure 7.

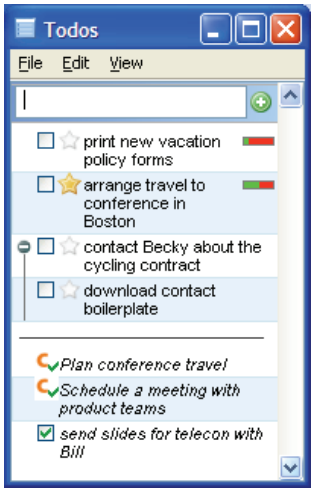

Figure 7. To Do list is automatically updated when the "Plan conference travel" action is completed.

Mapping and Interpreting To Do List Entries

Our approach progresses from syntactic to semantic interpretation, and leverages a number of broad-coverage knowledge sources. Our approach is informed by prior work in speech systems and dialogue systems [Wang & Acero, 2001, 2002; Seneff, 1992; Glass et al, 2004; Rich & Sidner 1997; Allen et al, 1995, 1996, 2001]. In particular, this work influenced the later stages of our processing, those concerned with mapping to semantic frames and task structures, while syntactic processing was incorporated as early processing to allow use of growing knowledge repositories and permit the system to function in the more dynamic learning conditions.

Figure 8 presents an overview of the stages that we use to map and interpret To Do list entries. There are seven distinct stages involved in this process.

The first stage is syntactic parsing. For this stage, we rely on Minipar, a popular dependency parser [Lin, 1998]. Minipar fit our needs because it is able to produce linkages when text is incomplete. Using the parser in the first stage occasionally presents some issues. For instance, a single-word entry "schedule" or "exercise" is interpreted as a noun, and the system is unable to interpret the entry, while longer entries such as "schedule a phone call" or "exercise on Tuesday" do get parsed appropriately.

The second stage is to identify semantic components in the user utterance. This stage includes identifying which strings express the action, the parameters, and the constraints in the To Do entry. We have developed a set of heuristics to process the somewhat idiosyncratic syntax of To Do entries. An entry such as "calculate costs for project" is processed into its semantic components: [calculate] [costs] [for project], consisting of the *action*, ("calculate") and the *phrases*, each containing an *entity*. The direct object of the action (eg. "costs" in the above example) is, for the sake of our processing, a phrase with the sole entity, cost. The entities are mapped to their basic form (for "costs", to the singular noun "cost") by the parser in the previous stage. The final component is a prepositional phrase, with the entity being "project". The entity is identified as the head noun of the phrase and its nominal (but not adjectival) modifiers, so given, for instance, the input phrase "for urgent sales meeting", the entity identified is "sales meeting".

The third stage is the lookup of known semantic components in the ontology. The ontology for actions consists of 85 tasks such as *Buy*, *Learn*, *Obtain Data*, *Print*. The action from the To Do list (eg, *reserve*), if found, is matched against these terms, trying both a direct match and a match using synonyms from WordNet (Fellbaum, 1990). The use of synonyms allows mapping of, e.g., the string "purchase" to the concept *Buy*. The ontology's set of concepts for entities is larger, containing more than 500 concepts, including *Breakfast*, *Bridge*, *Brunch*, *Building, Calendar, Chair, Contact Information*, etc. Again, alignment with these concepts is carried out by attempting both a literal and a synonym-based match.

The components which are in the execution assistant's ontology are identified. Semantic components which cannot be interpreted are passed for interpretation to the next stage.

The fourth stage is the semantic interpretation of out-of-ontology entities. This stage attempts to map terms not in the execution assistant's ontology to categories which are in the ontology. For instance, the ontology may not contain the concept of accounting package, but if the constantly learning, broad-coverage zero-cost repositories map "accounting package" to "software," then tasks such as "install accounting package" or "order accounting package" can be fully interpreted. Currently, only a small set of concepts is available to this stage, and we plan to use much larger repositories in this process. Similarly, if a new task is learned by the execution assistant, little may be known about it in the execution assistant's ontology, and this

stage can be of help in coping with mapping of additional concepts.

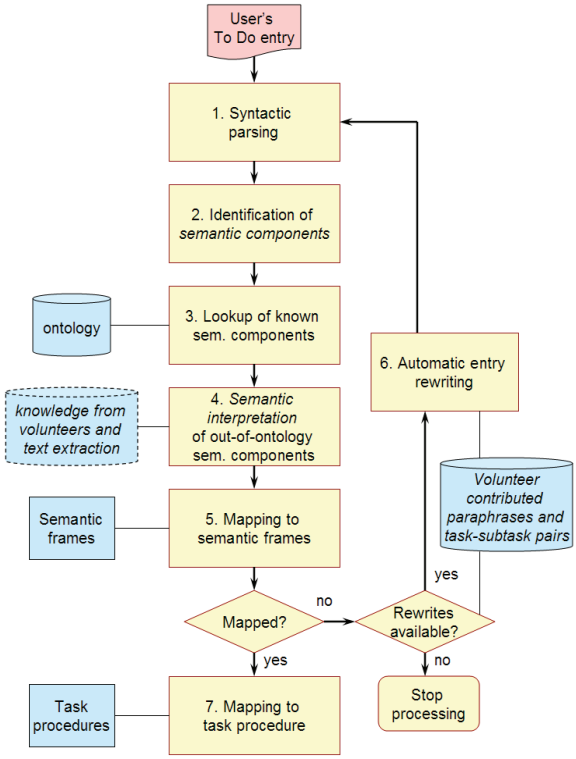

Figure 8. BEAM's To Do List Interpretation Steps and Knowledge Sources.

The fifth stage is the mapping to semantic frames, in which the semantic categories of the action and the entities are used to evoke the potentially matching semantic frames and see if the arguments are appropriate for this semantic frame.

If the statement could not be mapped to a semantic frame, the system checks if it is possible to rewrite the entry to a paraphrase or subtask, as explained below. If not, the system stops processing this statement. Recall that an important aspect of our domain is that not all entries may need to be interpreted by the system: the user may include entries not done at the office, such as picking up bread on the way home. However, if rewriting knowledge *is* available, the system attempts to interpret the statement by rewriting it.

The entry-rewriting knowledge in the current version of BEAM is constantly growing, and it currently allows rewriting of some actions to enable their interpretation. For instance the action "move" when the object is "meeting" is known as a paraphrase of "reschedule" a "meeting". A rewrite expression does not have to match literally – the entry being rewritten may have additional phrases, which are carried through.

Due to our method of collecting rewriting knowledge, we can in some cases be highly confident in the correctness and interpretability of the rewriting. In such cases, BEAM rewrites matching statements even before stage 1 of syntactic parsing. Such "aggressive rewriting" can help avoid incorrect semantic interpretation of the original statement. It can also serve as canonicalization of text before parsing it, for instance rewriting "w/" to "with".

If the mapping to semantic frame in stage five succeeded, BEAM proceeds to the seventh (final) stage of the mapping from semantic frames to task procedures. The semantic frames are aligned with task procedures. In mapping to task procedures, not that the user entry may not provide all the needed arguments, and additional interaction or learning may be required to identify the values of the unstated arguments.

Activating and Monitoring To Do Entries On Behalf of the User

The Procedure Catalog is BEAM's main entry point to the rest of the system. The catalog contains a description of all the tasks that other components in the system can accomplish for the user. It is organized as a task ontology, and a complete set of the parameters required for each task is given in terms of the system's ontologies of objects and classes. BEAM is integrated with the task ontology in the sense that it aims to produce mappings to any of the tasks described in it. However, we have focused to date on the kinds of tasks automated by the Spark execution and monitoring system [Morley & Myers, 2004]. Examples of these tasks are "Schedule a meeting," "Reschedule a meeting," "Create an agenda," "Plan conference travel," "Register for conference," and "Apply for reimbursement."

Once BEAM has mapped a To Do list entry, it notifies the user that it intends to act upon it to automate it. Then it sends the partially instantiated request to perform that task to Spark, which then asks the user for additional information and parameters that do not appear in the entry itself. BEAM also requests that Spark notifies it when the task is completed.

Spark can monitor the progress in completing a task once it is started. It has a Belief-Desire-Intention (BDI) architecture that is continuously sensing its environment and updating its beliefs accordingly. As a result, Spark can incorporate into its reasoning the effects of actions taken by other users or any other events registered in the system. This enables it to detect the completion of a task by external means, as well as suspending and canceling tasks that are no longer desirable. BEAM is notified when a task that it activates on behalf of the user is completed, suspended, or cancelled.

In the coming months we envision extending this integration more tightly with other components of the system that can accomplish tasks and that already have representations in the task ontology, including a calendar manager [Berry et al 2006] and an instrumented desktop [Cheyer et al 2005] that includes email parsing and sending capabilities. The task ontology continues to grow automatically as end users teach new tasks to the system or as the system learns on its own to automate repetitive tasks.

User Evaluations: Performance Watermarks

While we do not present conclusive experimental results in this paper, the overall system is undergoing a formal evaluation with several dozen users during September and October of 2006. The purpose of these evaluations is to test the learning capabilities of the overall system as it is used "in the wild" by several dozens of users. In the case of BEAM, the goal was to obtain initial data about the performance of the system, so we can compare it with next year's capabilities and demonstrate improvements. In fact, neither the knowledge repositories nor the Procedures Catalog are at the levels of content and coverage that we would want to see in order to conduct more conclusive evaluations of the effectiveness of the system. In any case, BEAM is integrated and already attempting to interpret and assist on every To Do list entry. It is fully instrumented, and all the data that is being collected will inform our future research.

BEAM is being evaluated in two experimental conditions. The first, BEAM-Baseline, does not use automatic rewriting and simply attempts to map To Do entries to the Procedures Catalog. The second, BEAM-Learning, has the functionality of the baseline system but it also performs automatic rewriting based on the paraphrasing knowledge that is learned from volunteers.

In the data that we have seen so far, BEAM was used on 440 statements. Without rewriting (BEAM-Baseline), a mapping was found 23.1% of the cases. With rewriting (BEAM-Learning), a mapping was found in 27.7% of the cases. Although this already shows improvement, in our case the purpose of the evaluations being conducted is to set the watermark based on the initial capabilities of BEAM and its integration within the overall system.

Conclusions and Future Work

Assisting users with To Do lists is a new and challenging research area for intelligent user interfaces. We outlined the various kinds of intelligent assistance obtainable through To Do list entry interpretation and automation, including acting on entries and reporting on their completion, adding new entries for anticipated subtasks, grouping and prioritizing entries, and coordinating To Do lists for joint activities across users. The central contribution of our work is an implemented system, BEAM, which processes To Do list entries and maps them to tasks that a system can automate for the user. The system then monitors the progress of the execution of those tasks and updates the To Do list when completed. Because To Do lists can cover many topics and tasks that may not be formalized in the underlying system, we use a novel approach to interpreting natural language tasks that exploits semi-formal knowledge repositories that are broad coverage and continuously increasing in size at zero cost. These repositories are populated with common knowledge by web volunteers and by automatic text extraction. BEAM has been fully integrated within a large system that implements various forms of assistance to users in office environments, and can execute and monitor To Do list tasks mapped by BEAM.

An important aspect of our research is that it has been heavily influenced by an office assistant architecture that learns to improve performance over time. As a result, learning is a central feature of our approach.

So far our work has focused on automating To Do list entries, but now that we have an implemented approach to interpret and map the To Do list entries we are very well positioned to undertake other kinds of assistance. We plan to extend BEAM to learn additional types of information about tasks and improve its capabilities to assist users to manage To Do lists. BEAM could anticipate "in-preparation" tasks and suggest them, mark tasks as user-only, and hypothesize groups of To Do entries that appear related. BEAM would improve its performance in all these tasks through collecting and exploiting additional types of common knowledge about tasks from web volunteers. So far, the task knowledge is collected and extracted off-line, but there is no reason why the system should not seek knowledge on-demand as it notices new and unknown To Do list items.

We discussed earlier in detail the new challenges that we face with respect to existing approaches in natural language dialogue and speech systems, as we build on their work on semantic grammars. An aspect we did not discuss is that 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]. The overall task structure is represented formally in various forms: scripts, task hierarchies, or intentional structures (SharedPlans). The goal of the system is to complete as much of the task requirements as the prior dialogue permits. To date, we treat To Do entries as not related and their progress is monitored separately. Tracking dialogue over time and overall task progress have not been an issue, but we noted grouping of tasks as a challenge early in the paper. The approaches developed for dialogue and collaborative systems would provide a valuable framework for work in this direction.

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.

We would also like to incorporate useful design criteria suggested by user studies in cognitive psychology and human factors regarding To Do lists. 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 DARPA under contract no. NBCHD030010. We thank Karen Myers and Ken Conley for their feedback on this work and their support in integrating BEAM with the PExA project within the CALO architecture.

References

1. Allen, J.F. et al, The TRAINS Project: A Case Study in Defining a Conversational Planning Agent, Journal of Experimental and Theoretical AI, 1995.
2. Allen, J.F., B. Miller, E. Ringger and T. Sikorski A Robust System for Natural Spoken Dialogue, Proc. 34th *Assoc. for Computational Linguistics (ACL)*, 1996.
3. J. Allen, G. Ferguson, and A. Stent. 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.
4. Bellotti, V.; Ducheneaut, N.; Howard, M., Smith, I. Taking email to task: the design and evaluation of a task management centered email tool. ACM Conference on Human Factors in Computing Systems (CHI 2003); NY: ACM; 2003; 345-352.
5. Bellotti, V., B. Dalal, N. Good, P. Flynn, D. Bobrow, N. Ducheneaut. What a To-Do: Studies of Task Management Towards the Design of a Personal Task List Manager. *CHI 2004*
6. Berry, P.; Conley, K.; Gervasio, M.; Peintner, B.; Uribe, T.; and Yorke-Smith, N. Deploying a Personalized Time Management Agent. *Proceedings of AAMAS'06* Industrial Track, Hakodate, Japan, 2006.
7. Blythe, J. Task Learning by Instruction in Tailor. *In Proc. of Intelligent User Interfaces (IUI-2005)*, 2005
8. Cheyer, A. and Park, J. and Giuli, R. IRIS: Integrate. Relate. Infer. Share. *1st Workshop on The Semantic Desktop. ISWC* 2005.
9. 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-05)*
10. T. Chklovski. 2005. Collecting Paraphrase Corpora from Volunteer Contributors. In *Proceedings of the Third International Conference on Knowledge Capture (K-CAP 2005)*. 2005
11. T. Chklovski, 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-04)*.
12. T. Chklovski, 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, 2005
13. T. Chklovski. LEARNER: A System for Acquiring Commonsense Knowledge by Analogy. *Proceedings of Second International Conference on Knowledge Capture (K-CAP 2003)*. 2003.
14. G. Chung, S. Seneff, and C. Wang, "Automatic Induction of Language Model Data for a Spoken Dialogue System," *Proc. SIGDIAL*, 2005.
15. A Dey, G Abowd. CybreMinder: A Context-Aware System for Supporting Reminders. HUC, 2000
16. Fillmore, C.J.: ''The Case for Case,'' in Bach and Harms (Eds.), *Universals in Linguistic Theory*, 1969
17. M. Gavalda and A. Waibel. Growing semantic grammars. In Proceedings of the COLING/ACL, Montreal, Canada, 1998.
18. J. Glass, E. Weinstein, S. Cyphers, J. Polifroni, G. Chung, and M. Nakano, "A Framework for Developing Conversational User Interfaces," Proc. *CADUI*, 354-365, Funchal, Portugal, January 2004.
19. Horvitz, E., A. Jacobs, D. Hovel. Attention-Sensitive Alerting, *Proceedings of UAI '99, Conference on Uncertainty and Artificial Intelligence*, July 1999, Morgan Kaufmann Publishers: San Francisco. pp. 305-313.
20. Jones, S. R. and P. J. Thomas. "Empirical assessment of individuals' 'personal information management systems'." *Behaviour & Information Technology* 16(3): 158-160. 1997
21. Harrison, B. An Activity-Centric Approach To Context-Sensitive Time Management. In *CHI 2004: Workshop on the Temporal Aspects of Work*.
22. Lin, D. Dependency-based evaluation of minipar, In Proceedings of *Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*. 1998.
23. 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.
24. Miller, G. A., Ed. WordNet: An on-line lexical database. International Journal of Lexicography 3, 4 (Winter 1990), 235-312
25. Mitchell, T., Wang, S., Huang, Y., and Cheyer, A. Extracting Knowledge about Users' Activities from Raw Workstation Contents. *AAAI* 2006.
26. Morley, D., and Myers, K. 2004. The SPARK agent framework. In *Proc. of AAMAS'04*, 714–721.
27. Myers., K., P. Berry, J. Blythe, K. Conley, M. Gervasio, D. McGuinness, D. Morley, A. Pfeffer, M. Pollack, M. Tambe. An Intelligent Personal Assistant for Task and Time Management. *AI Magazine*, to appear.
28. Norman, D. A. Cognitive artifacts. In J. Carroll, editor, Designing Interaction: Psychology at the Human-Computer Interface, pages 17--38. Cambridge University Press, New York, NY, USA, 1991.
29. Rich, C. and C. Sidner. COLLAGEN: When agents collaborate with people. *In Proceedings of the International Conference on Autonomous Agents (Agents'97)*, 1997.
30. Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., Oh, A. Creating natural dialogs in the Carnegie Mellon Communicator system. *Proceedings of Eurospeech*, 1999, 4, 1531-1534
31. Seneff, S. "TINA: A natural language system for spoken language applications," *Computational Linguistics*, 18(1), 1992.
32. 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.
33. 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.
34. Y. Wang and A. Acero. Grammar Learning for Spoken Language Understanding, in Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding. Madonna di Campiglio, Italy, 2001.
35. Y. Wang, A. Acero. Evaluation of Spoken Language Grammar Learning in the ATIS Domain, in Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing. Orlando, Florida, May, 2002.
36. W. Ward. *Understanding Spontaneous Speech: the PHOENIX System*. In Proceedings International Conference on Acoustics, Speech, and Signal Processing, pages 365-68, Toronto, 1991.