"Where Are the Christmas Decorations?" : A Memory Assistant for Storage Locations

Lewis Creary and Michael VanHilst

Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, California
(creary,vanhilst)@hpl.hp.com

Abstract. At Hewlett-Packard Laboratories we want to know how inexpensive it can be to endow mobile personal assistants with the ability to speak naturally with their users. To this end, we are investigating and demonstrating speech-capable mobile personal assistants that can be realized using common off-the-shelf hardware and software components, a modest development effort, and wireless connectivity. As a part of this activity, we built a storage location memory assistant by connecting a voice front end on a PDA to a database back end on a remote server, with speech, natural language, and dialog processing in between. Specifically, the storage location memory assistant saves and retrieves information about the locations of stored objects in and around the user's house. Here is a sample dialog:

User: *Where are the Christmas decorations?*
PDA: *They're in the leftmost medium-sized white box under the wood table in the garage.*

To implement the storage location memory assistant, we leveraged technologies in speech recording, voice recognition, parser generation, and database management to produce domain-limited natural language understanding and semi-structured knowledge representation. In this paper, we describe the architecture of the system, and some of the more interesting technical details.

Keywords: Natural language understanding; Database systems; Semi-structured data; Speech recognition

1 Introduction

A memory assistant helps a user remember things and events. Some electronic memory assistants – notably, note pads, appointment calendars, and address books – are already common in everyday use. A number of memory assistants also appear in the research literature [La94], [LF94], [RS96]. At Hewlett-Packard Laboratories, we built a prototype memory assistant with a spoken natural language interface. We wanted to know how much effort it would require, using today's off-the-shelf technology, to build a personal digital assistant that communicates by speaking and being spoken to, and accepts information expressed in natural, everyday language. Ultimately, we hope to investigate the usefulness of such devices.

Hewlett-Packard's CEO, Carly Fiorina, recently asked the company's employees to "return to the garage" in search of new ideas. One of the authors, who spends most of his time in the garage looking for things that are stored there, thought that it would be valuable to have a device that could be told, and later recall, the locations of items in the garage. Accordingly, we decided to implement a memory assistant for storage locations.

A user communicates with our storage location memory assistant by speaking to a PDA that has a built-in microphone. The application receives a recorded version of the user's communication, transcribes it with speech recognition software, parses the resulting text, determines its meaning, and finally performs appropriate operations (queries and/or stores) on a relational database. An appropriate response is generated and synthesized into an audio speech file that is then played back to the user on the PDA. Our PDA has a wireless LAN card, allowing the user to walk around freely within a home or office environment.

Prior research memory assistants, from Xerox [La94], [LF94] and MIT [Ad99], [RS96], focused on automatically collecting information without user involvement. By contrast, we hope to explore whether the use of natural language and speech can make interaction simple enough that a user would be willing to participate in the process of collecting information. (None of the prior systems had a natural language interface.)

In the next section we describe some of the things the memory assistant can do. In Sections 3 and 4 we discuss, respectively, the architecture of the memory assistant and the speech recognition technology that it uses. In Sections 5 and 6 we describe the memory assistant's representation of stored knowledge, and the natural language and storage-related processing that it performs. Finally, in Sections 7 and 8, we discuss outstanding issues and future directions of our research, and formulate the overall conclusion that we have (so far) drawn from it.

2 Features of the Memory Assistant

Below is a sample of a dialog between a user and the memory assistant. Through earlier conversation with the user, the memory assistant has already learned about a house with its various rooms and garage, and about a collection of objects located in the garage. The objects include a wooden table, a green desk, a metal table on the desk, two plastic buckets (one on and one below the metal table), and a bag containing a PC on the floor next to the desk. In the dialog, sentences in parentheses are displayed by the memory assistant in a text window, but not spoken. Between sections of displayed dialog, we comment on features of the memory assistant that the preceding section of dialog illustrates.

User: *Where is Herb's old PC?*
PDA: *It's in the orange bag on the floor by the green desk in the garage.*
User: *I put Lew's tennis shoes in the top left drawer of the desk.*

PDA: *Sorry, but I couldn't find any landmark object to anchor your location description. Please try again.*
User: *I put Lew's tennis shoes in the top left drawer of the desk in the garage.*
PDA: (I'm inferring the existence of a previously unknown drawer.)
PDA: *Ok, got that.*

The application requires that every object be located in a chain of locations anchored to some user-specified landmark object in the context of the user's current location. In the above example, the user's location context is global (the house). The garage is a global landmark, but the desk is not. The desk is, however, a local landmark when the location context is the garage, as the next example illustrates.

User: *I'm in the garage.*
PDA: (User location context is now the garage.)
User: *Where are the paint brushes?*
PDA: *They're in a white plastic bucket on the metal table on the green desk.*

Since the user location context is now the garage, the memory assistant no longer mentions the garage explicitly in location descriptions that involve local (i.e., garage-relative) landmarks. The user also does not need to mention the garage when describing the location of an item in that context. (In the future, the need to declare the user's location context could be replaced by a location sensor on the PDA.)

User: *The tax records are under the college textbooks in the leftmost brown box on the wooden table.*
PDA: (I'm inferring the existence of a previously unknown box.)
PDA: (I'm inferring the existence of a previously unknown group of textbooks.)
PDA: *Ok, got that.*
User: *Where are the college textbooks?*
PDA: *They're in the leftmost brown box on the wood table.*
User: *Where is the box containing college textbooks?*
PDA: *It's on the wood table.*

When given a chain of objects in a description, the application learns not only the object at the beginning of the chain, but also any other objects within the chain that were not known before. In the above example, the application learned of the college textbooks and a new box from the user's description of the location of the tax records.

Some objects that contain other objects can be identified not only by their properties (as discussed later in this section), but also by the objects they contain. In the above example, the box containing the college textbooks has no described properties other than being a box. But it could still be identified as the box *containing* college textbooks.

User: *There are 20 pencils in the center drawer of the desk.*
PDA: (I'm inferring the existence of a previously unknown drawer.)
PDA: *Ok, got that.*
User: *Where are the 20 pencils?*

PDA: *They're in the center drawer of the green desk.*
User: *Where is the drawer containing 20 pencils?*
PDA: *It's in the green desk.*

The memory assistant includes special treatment for groups of objects. A group can have a number as its size, and will be treated as a plural in a response. Pants are considered a group, as are the tennis shoes mentioned earlier.

User: *I'm moving the box containing college textbooks underneath the table in the master bedroom.*
PDA: *Ok, got that.*
User: *Where are the tax records?*
PDA: *They're under the college textbooks in the brown box under the table in Herb's large master bedroom.*

The Memory Assistant's description of an object in a response to the user includes all of the object's properties. However, the user does not need to include all of the properties in referring to the object – only enough to uniquely identify it. In the above example, there is only one master bedroom, but it has the additional properties of being large, and belonging to Herb.

When the memory assistant generates a response containing a noun phrase, the property-words used in the noun phrase are ordered according to their corresponding attributes, so as to produce a natural sounding sentence. For example, the ordering puts size-words before color-words, with type-words coming at the end, just before the noun (e.g., "large gray wading boots".) The ordering used is specific to English, and would, of course, need to be changed if some other language were being used.

User: *I'm putting a sponge under the counter in the kitchen.*
PDA: (I'm inferring the existence of a previously unknown counter.)
PDA: *Ok, got that.*
User: *Where is the sponge?*
PDA: *There are 2 things matching that description: First, a large blue sponge is in the white plastic bucket on the white shelf under the metal table on the green desk. Second, a sponge is under the counter in the kitchen.*
User: *The second one is yellow.*
PDA: *Ok, got that.*
User: *Where is the yellow sponge?*
PDA: *It's under the counter in the kitchen.*

The last example illustrates two important capabilities of the memory assistant – the ability to refer to objects that cannot be uniquely identified by their properties, and the ability to augment the description of known objects. When an object is added, the properties of either the new object or a previously stored object may be insufficient to distinguish one of the objects from the other. To handle this situation, the application remembers the enumerated list of possible referents from the previous response. The user's description in terms of the enumeration then becomes a valid object identifier.

3 Architecture

The storage location memory assistant is implemented in six modules. Fig. 1 shows the modules in this architecture. Most of the pieces are general purpose. The Speech Transcriber, Speech Synthesizer, and Database modules are simple wrappers around inexpensive off-the-shelf software components. The PDA module is concerned with recording and playing voice files, while the Control module caches and coordinates the movement of messages. Only the Natural Language module contains code specific to the location memory task. By comparison, the MIT Galaxy system [Se98] distributes task-specific knowledge among many custom components. Galaxy applications are correspondingly more difficult to craft.

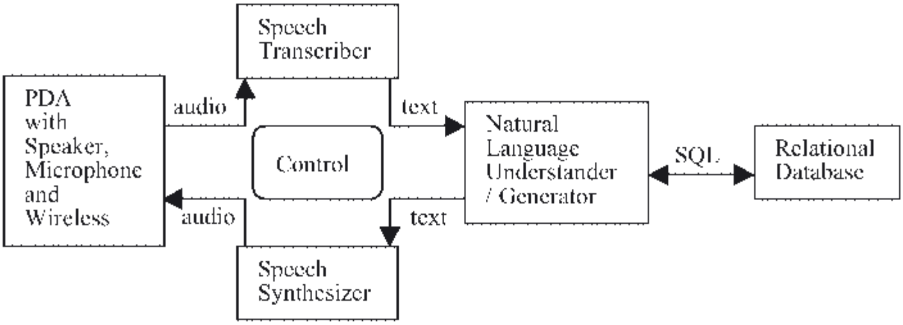

Fig. 1. The architecture of the storage location memory assistant.

Each module runs in its own process, possibly on a separate machine. Communication among the modules is asynchronous and coordinated by the Control module. The use of asynchronous message passing allows the Natural Language module to generate multiple responses to a single input message. It also allows the user to enter several statements without waiting for a response to each one (i.e., when telling the memory assistant about the locations of things). Messages are queued by the Control module while their intended recipients are busy. The Control module also caches some messages after they have been consumed, allowing the user to ask about messages that were missed or misunderstood. (To support this interaction, the controller itself understands certain simple messages coming out of the Speech Transcriber).

The specific off-the-shelf components are meant to be more-or-less interchangeable. Our current implementation uses an HP Jornada (running Windows CE) as the PDA device with an 802.11b card for wireless connectivity. The speech recognition and transcription is performed by a version of Dragon Naturally Speaking, while we use Via Voice for the speech synthesis. The data is stored in a MySQL database accessed through ODBC.

4 Speech Recognition

Speech transcription is handled by a relatively inexpensive voice dictation system, of the type widely available in stores for voice enabling common tasks and dictating documents. Common speech engines, like Dragon Naturally Speaking and IBM Via Voice have developer friendly APIs that allow them to be easily combined with custom application code. Using a general dictation speech recognizer had some advantages and some disadvantages. The main advantage was that we could accept fairly natural language. The main disadvantage, at least with today's technology, was the need to generate a corpus of sample input sentences on which to train the recognizer.

We investigated a number of alternative technologies before settling on a general dictation engine. Although our domain is restricted, our natural language module understands a wide range of utterances within that domain. We needed a speech recognizer that would recognize and accept the same utterances. Menu or word list types of grammars, commonly used for voice applications, would have limited acceptable word sequences to an unnatural extent. Since our natural language parsing uses a context-free grammar, it might have seemed that a speech recognizer that accepted a CFG language specification would be appropriate. But, of the two we tried, one ran slowly and gave poor results, while the other used the CFG as a yes/no filter after recognition, rather than as the model for guiding recognition. We concluded that CFG based speech recognition could be acceptable for simple grammars with few rules, but were not easily adapted to applications requiring a rich grammar, such as our own.

Dictation engines use n-gram language models, where short sequences of consecutive words are assigned different levels of probability. Commercial products come with an initial general-language model of probabilities. The language model can be refined to achieve higher accuracy in a given domain by analyzing text samples specific to that domain. We trained our recognizer on a corpus of random sentences composed from different adjective groups, noun groups, and location groups specific to our grammar. We then untrained pathological errors that appeared in early trials.

At this time, the application works fairly well. Although recognition is still not 100%, it is consistent enough to give demonstrations with few or no recognition failures. In the future we expect the technology to improve to where less effort is required to achieve higher accuracy. We may also try a higher-end speech recognition system, rather than the low-cost software we are currently using.

5 Representation of Stored Knowledge

The memory assistant's "knowledge" is stored in a relational database. There are only two tables, Attributes and Locations, each with a simple schema. The Attributes table stores properties of single objects, while the Locations table stores locative relationships between pairs of objects. An important feature of our storage model is that no distinction is made between objects and places. One could as well be looking for a yellow box as placing items in it, and any item can be beside another item or have other items on top of it.

Properties (such as 'large', 'green', 'wooden', etc.) are used to characterize objects in ways that are useful and natural for the user, and to distinguish one object from another. Attributes are used to collect the various properties into semantically significant families, such as relative size, color, and material. The only required attributes are 'class' and 'object identifier'. The former specifies what basic kind of thing an object is, such as a table, book, or sponge, while the latter specifies a unique internal identifier for use in database tables. All other attributes are optional. There is a type attribute, which is used to represent the extra-class content of compound nouns, such as 'living room', 'ski mask', and 'bottle opener'. Other attributes include relative age ('old' or 'new'), owner, and relative vertical and horizontal positions (e.g., 'lower' and 'left'). The attributes table has three fields, Object-ID, Attribute, and Value. Thus far, only single-valued attributes have been found necessary.

The location of an item is determined by a chain of relationships starting with the item in question, and ending with an item or place that has a special status as a *landmark*. A landmark is an object that the user has designated as a satisfactory terminus of an answer to a "Where is …" query. Landmark status may be either global or local. A *global* landmark has landmark status unconditionally (e.g., the garage), whereas a *local* landmark can function as a landmark only when the "user location" aspect of the conversational context has a certain value (e.g., a certain desk might have landmark status only when the user location context is the garage).

The memory assistant allows a variety of locative relationships, including *in*, *on*, *under*, *beside*, and *of* (i.e., *part-of*, as in "the top drawer of the desk"). The NL understander allows some additional relation-terms, such as 'alongside' or 'into', which it normalizes to one of the canonical relation-terms used by the storage module. To give an example of a chain of objects locating a given object, a book might be "on the lower shelf of the closet in the living room." Here, lower shelf and closet are not unique in the house (there are other lower shelves and other closets). But, in this example, there is only one living room, and it has been given global landmark status.

The fields in the Locations table are Relation, Object-ID1, and Object-ID2. In the future, we plan to add a Time field to record when an object was reported in that location. This will allow the system to maintain location histories of objects, and to report the age of information – which may help the user in assessing its reliability.

Our data representation is similar to that used in other systems for storing semi-structured information, such as Stanford's Lore [Ab97], [McH97] and MIT's Haystack [Ad99]. Lore's representation is more general than ours, making it easier, for example, to add properties of properties (like the Time field mentioned above). Haystack supports additional semantic meaning for the relationships in order to facilitate data navigation. Data navigation is not an issue in our application.

6 Natural Language and Storage-Related Processing

In general, detailed computer understanding of unrestricted English text is an extremely difficult task, raising many issues that are still not well understood in the computational linguistics community. However, the task is much more tractable if the vocabulary and linguistic constructions involved are limited to those necessary for dealing with a particular well-defined task domain, such as that of the storage location memory assistant. Part of the research agenda for this project is to investigate the extent to which natural, domain-specific English understanding capabilities can be constructed for particular applications using readily available computational tools.

Our memory assistant processes sentences in three phases: semantic interpretation, information storage/retrieval, and response generation. First, during bottom-up parsing a semantic representation is computed compositionally for each phrase, from the semantic representations of the phrase's constituents. This assigns to the sentence itself a semantic interpretation of the form *<sa, dc>*, where *sa* is a *speech act* label (such as 'locative assertion' or 'locative query'), and *dc* is a *descriptive content* expression describing the properties of objects and/or relations among objects. In such expressions, individual objects are represented by attribute-value pairs, and relations among objects are represented by quasi-logical formulas.

In the second processing phase, a sentential semantic representation $<sa_i, dc_i>$ is analyzed and processed, usually by storing and/or retrieving appropriate pieces of information. Depending on the content of $<sa_i, dc_i>$, this processing will be more or less complex. For example, in the case of an assertion that Lew's tennis shoes are in the top left drawer of the desk in the garage, the memory assistant will first check to see whether the database has entries for a unique desk in the garage, and for a top left drawer of that desk. Supposing that it finds an entry for the desk but not for the drawer, the memory assistant will infer the presence of the drawer, report this inference to the user, put a new entry for the top left drawer into the database, add a new database entry for the tennis shoes, and finally issue a "Got that" acknowledgment to the user. At a lower level of detail, this processing will involve, among other things, the performance of some diagnostic reasoning, the generation and execution of several custom SQL statements, and the making of several calls on a user response module.

The third processing phase (response generation) is often interleaved with second-phase processing, as in the example given in the previous paragraph. Also as in that example, third-phase processing may be quite simple, involving only the generation of fixed or parameterized expressions. However, generating a response to a location query is somewhat more complex, and involves choosing the appropriate article (*a*, *an*, or *the*) for each item in a chain of location objects, and ordering properly the adjectives and other modifiers that describe each object in the chain.

To get some idea of the particular representations and queries involved in the processing just described, consider the question: *Where are the Christmas decorations?* For this sentence the semantic-interpretation processing phase yields the semantic representation:

```
<LOC_QUERY, "ObjID    GRP_DECORATION_2
             Class    Group
             GrpClass Decoration
             Type     Christmas">.
```

The information-storage/retrieval processing phase involves “on-the-fly” construction of the SQL query:

```
SELECT A1.ObjID
  FROM Attributes AS A1, Attributes AS A2, Attributes AS A3
 WHERE A1.ObjID = A2.ObjID       AND A1.ObjID = A3.ObjID
   AND A1.Attribute = ‘Class’    AND A1.Value = ‘Group’
   AND A2.Attribute = ‘GrpClass’ AND A2.Value = ‘Decoration’
   AND A3.Attribute = ‘Type’     AND A3.Value = ‘Christmas’.
```

This query is used to determine the object identifier of the Christmas decorations group, which is then used, together with repeated SQL queries of the form

```
SELECT Relation, ObjID2 FROM Locations WHERE ObjID1 = <objid>,
```

to find three <relation, object identifier> pairs for a containing box, an associated table, and the garage. This sequence of pairs is then used to generate the final response: *They're in the leftmost medium-sized white box under the wood table in the garage.*

The Natural Language Understander/Generator is written in Perl supplemented with yapp, a yacc-like LALR(1) parser generator. The input to yapp is a context-free grammar with 50 rules. The lexicon presently contains 265 single words, 95 compound nouns combining some of those single words, and several variants each of a half-dozen or so fixed word sequences. The size of the grammar and lexicon reflect the limited nature of the domain. There are several kinds of statement possible, including those that report an item's properties and location, those that request an item's location, and those that report a change in an item's location. The domain is verb-poor and noun-rich. Most of the complexity in the grammar is devoted to describing the various types of noun phrases recognized by the memory assistant.

7 Issues and Future Work

Experience with the location memory assistant has revealed several outstanding issues. For example, when the user says, *I am putting the Christmas decorations in a box on the shelf*, is that a previously unknown box or one that is already represented in the database? We try to infer as much information as possible from each statement, so that the user is not forced to use rigid and stylized sequences of statements when introducing new information. But we have not found a way of distinguishing old and new items that is both simple and robust. For example, we cannot assume that the user will always use a definite determiner when referring to an object that is already known to the memory assistant, because the user may not remember which objects the memory assistant has been told about. Nor, for the same reason, can we assume that the user will always use an indefinite determiner in referring to an object that is new to the memory assistant. In general, what is needed here is to give the memory assistant a set of procedures and heuristics that are usually successful in distinguishing old and new items, and also make it easy for the user to discover and repair any discrepancies that may arise between the real world and the memory assistant's model of the world.

There is currently no mechanism for the user to add new attributes and attribute families. Certainly, we cannot know ahead of time all the types of things a user might want to store or all the ways in which a user might wish to identify them. The solution is not as simple as adding terms to a general-purpose dictionary. The Natural Language module requires that several specific kinds of knowledge be provided about each new word added, such as what attribute (semantic family of properties) a new adjective should be associated with, whether a new noun has an irregular plural, what other words could be used with a new noun to form compound nouns, whether a new word can be a part of one or more compound nouns, whether a new noun could also function as an adjective (as the word 'glass' can), and whether a new word is merely a synonym of a word already in the lexicon. Also, the user may need to introduce a new attribute, such as hardness or brand-name. We believe that allowing custom user extensions to the domain vocabulary would require a special tool that incorporates knowledge about the different kinds of word-information needed by the Natural Language module, and is prepared to guide the user through a vocabulary augmentation dialog.

Ultimately, we would like to extend the domain to include larger collections of objects and properties, and also to accommodate other contexts, i.e., an office environment. We expect that, in attempting such extensions, we will encounter other challenges involving both language understanding and knowledge representation.

8 Conclusion

From our experience in this project we conclude that, by leveraging off-the-shelf hardware and software technologies and working within a restricted domain, it is indeed possible to construct mobile personal assistants that are interesting and useful with a modest development effort. Common off-the-shelf speech recognizers and synthesizers are reaching the level of genuine usability for applications such as our own, while a standard parser generator can handle the relatively small grammar covering the English constructions needed for natural communication in our limited domain. We expect that, with the rapid proliferation of portable hand-held devices with capabilities like the Jornada used in our demonstrations, more applications like the one we describe here will begin to appear and, ultimately, enter everyday use.

Acknowledgments

We have benefited from useful comments by Reed Letsinger, Bernard Burg, and two anonymous reviewers on earlier drafts of this paper.

Bibliography

[Ab97] Abiteboul, S., Querying Semi-Structured Data, In Proceedings of the International Conference on Database Theory (ICDT), Delphi, Greece, January 1997, pp. 1—18

[Ad99] Adar, E., et. al., Haystack: Per-User Information Environments, In Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99), November 2-6, 1999, Kansas City, Missouri

[La94] Lamming, M., et. al., The Design of a Human Memory Prosthesis, The Computer Journal, volume 37, number 3, 1994

[LF94] Lamming, M. & Flynn, M., "Forget-me-not": Intimate Computing in Support of Human Memory, Rank Xerox Research Center Technical Report EPC-1994-103

[McH97] McHugh, J., et. al., Lore: A Database Management System for Semistructured Data, SIGMOD Record, 26(3):54—66, September 1997

[RS96] Rhodes, J.R. & Starner, T., Remembrance Agent: A continuously Running Automated Information Retrieval System, In Proceedings of the First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology (PAAM'96), pp. 487—495

[Se98] Seneff, S., et. al., Galaxy-II: A Reference Architecture for Conversational System Development. In Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP), Sydney, Australia, November 1998, pp. 931—934.