CARDIAC: An Intelligent Conversational Assistant for Chronic Heart Failure Patient Heath Monitoring

George Ferguson[1], James Allen[1,2], Lucian Galescu[2], Jill Quinn[3], Mary Swift[1]

[1] Department of Computer Science, University of Rochester, Rochester, NY 14627
[2] Institute for Human and Machine Cognition, 40 South Alcaniz Street Pensacola, FL 32502
[3] School of Nursing, University of Rochester, 601 Elmwood Avenue, Rochester NY 14642
{ferguson, james, swift}@cs.rochester.edu, galescu@ihmc.us, jill_quinn@urmc.rochester.edu

Abstract

We describe CARDIAC, a prototype for an intelligent conversational assistant that provides health monitoring for chronic heart failure patients. CARDIAC supports user initiative through its ability to understand natural language and connect it to intention recognition. The natural language interface allows patients to interact with CARDIAC without special training. The system is designed to understand information that arises spontaneously in the course of the interview. If the patient gives more detail than necessary for answering a question, the system updates the user model accordingly. CARDIAC is a first step towards developing cost-effective, customizable, automated in-home conversational assistants that help patients manage their care and monitor their health using natural language.

Introduction

Research suggests that the quality of health care is greatly affected by the support patients receive in the home, whether by family caregivers or from providers such as nurse practitioners. For example, readmission rates for patients who have experienced congestive heart failure can be significantly reduced with close home monitoring of patients by a nurse practitioner (Naylor *et al*. 2004). Given the number of chronic heart failure patients, however, there is not enough medical personnel available to provide such close monitoring, and even if there was the cost would be prohibitive. New technologies for health monitoring are being implemented, but they can be complex and time-consuming for people to use. And, these monitoring devices allow limited input for the patient's subjective health status report. The focus of our research is to develop an assistant for chronic care patients to help them look after themselves and provide comprehensive health care monitoring.

In this paper we give an overview of CARDIAC (Computer Assistant for Robust Dialogue Interaction and Care), a prototype of an intelligent conversational assistant that provides health monitoring for chronic heart failure patients. CARDIAC's objective is to conduct regular "checkup" interviews with patients to collect information relevant to their condition. The target population for CARDIAC is patients who are at home following specific self-care guidelines to manage their heart failure condition. The CARDIAC checkup is designed to obtain the information required by the self-care guidelines including both objective (*e.g.*, weight) and subjective (*e.g.*, pain) aspects of their condition. This is a first step towards providing automated decision support based on the guidelines. The system can also take advantage of other sources of information (such as a network-connected scale or, ultimately, a PHR) and use it effectively in the checkup without additional programming.

The system's conversational interface is intuitive and easy to use, a benefit that may encourage patients to report their information more often. This system is a first step towards our long-term goal of developing cost-effective, customizable, automated in-home conversational assistants that help patients manage their treatment and monitor their health using natural spoken dialogue over the telephone or with in-home systems.

Overview of the approach

CARDIAC is an agent-based spoken dialogue system that conducts health monitoring interviews with chronic heart failure patients using natural language. CARDIAC interprets the patient information and uses it to update its user models. The following dialogue excerpt illustrates the sort of interaction that the system supports.

SYS: Do you know your weight
USR: YES
SYS: Got it
SYS: What is your weight
USR: TWO SIXTY

Proceedings of the AAAI Fall Symposium on Virtual Healthcare Interaction, Arlington, VA, November, 2009.

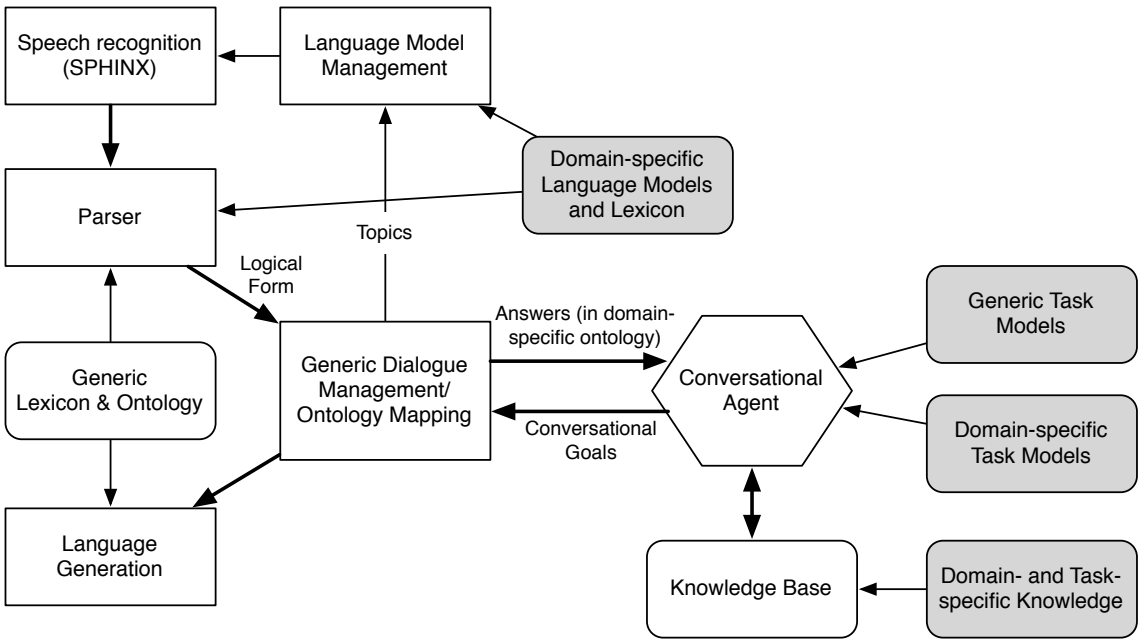

Figure 2: CARDIAC System Architecture

SYS: Did you say 260
USR: YES
SYS: Do you have shortness of breath today
USR: YES
SYS: How severe is the shortness of breath
USR: A LITTLE MORE THAN NORMAL

Figure 1 Sample dialogue excerpt.

The CARDIAC system is built as an instantiation of the TRIPS generic dialogue architecture (Allen *et al*. 2000), which provides broad coverage domain-general parsing, generation, and discourse interpretation facilities. To adapt this generic system to a specific domain, we need to define a task model that the conversational agent uses to drive the interview process without needing to manage the details of natural conversation. Besides the task model, other domain specific components needed are the domain specific lexicon (in this case, medication names) and statistical language models used in speech recognition and generation. In addition, we need ontology mapping rules that translate meanings expressed in the TRIPS generic ontology to the specific ontology used for reasoning in this domain.

The system architecture is given in Figure 2, with the generic components in white and the domain specific components in grey. To obtain the most accurate speech recognition (SR) we can get, we use dynamic language models based on the current topic. The topic is determined when the system asks a specific question, at which point the SR engine switches the language model accordingly. The best SR hypothesis is used as input to the parser. The interpretation manager (IM) receives the parser output and performs contextual interpretation including reference resolution and interpreting elliptical answers. The IM and the Conversational Agent (CA) interact to perform goal-driven dialogue management; *i.e*., the system is driven by reasoning about goals to acquire knowledge. Each of these components is described in more detail below.

Speech Recognition

Because no domain-specific textual data was available to train statistical language models for speech recognition, we used the technique described in (Galescu *et al*. 1998). In short, this technique allows us to build quickly language models using a process of collecting domain-specific utterances, then generalizing them (via synonyms and rephrasing) into a finite-state grammar from which we generate a large corpus of sentences; statistical language models are trained on this "artificial" sentence corpus. For the CARDIAC system we trained a number of topic-specific grammars and language models; topics were based on broad categories, such as symptoms (*e.g*., fatigue, edema, etc.), medication use, diet and exercise. However, because we expect that in reality the system would likely be faced with off-topic over-answers (*e.g*., "*No swelling, but I was very short of breath yesterday after my daily exercise*"), in the final system all the topic-specific models were interpolated, with most of the emphasis on the main topic -- this way we traded off some specificity for greater robustness. The interpolation weights were tuned to some extent using data collected from simulated patients. Preliminary data suggests that, on average, using dynamic LMs may improve performance by anywhere from 2% to 5% (relative) compared to using a static LM.

Parsing

The best SR hypothesis is then fed to a deep, broad coverage parser. The parser uses a general grammar and a domain-independent semantic lexicon augmented by domain-specific words, which in this system are medication names, to produce a semantic representation of the utterance. In order to deal with speech recognition errors, the parser is designed to find the most semantically coherent sequence of sentence fragments when it cannot construct a complete analysis.

Interpretation Manager

The IM receives the parser output and performs contextual interpretation including reference resolution, surface speech act recognition, and interpretation elliptical answers. It generates possible hypotheses about the user's intended meaning, which then are passed to the conversational agent (CA) for evaluation as to whether the hypothesis would be a coherent statement. After evaluating the promising hypotheses, the IM either chooses one as the final interpretation and passes it to the conversational agent, or determines that it didn't understand the user's response. Often, because of recognition errors, some part of the utterance is not interpretable. If the IM can identify a fragment that satisfies the goal (to the CA's satisfaction), then it ignores the fragments that cannot be interpreted. While the conversational agent determines the current goal for the interview (*e.g.*, find out the patient's weight), the IM manages the details of the actual dialogue. Thus, if the user's answer is not understood, it will re-ask the question, possibly also giving hints such as asking the patient to speak more simply. In addition, when dealing with answers that could easily be misrecognized, such as answers involving numbers, the IM may initiate a confirmation subdialogue to verify the answer. If the system fails to understand the patient a number of times, the IM abandons its discourse-level goal and notifies the conversational agent to abandon the current task-level goal. The IM also applies ontology mapping rules to convert the generic semantic representation output by the parser into the domain-specific representation whenever it communicates with the CA.

Conversational Agent

The CA is responsible for the system's overall behavior. This includes the following responsibilities:

- It is driven by a declarative model of the task(s) that the system can perform. Based on these tasks, it manages the goals that drive the system's behavior.
- During execution, it maintains the knowledge base that stores what the system knows about the current situation.
- It interacts with the language understanding components to support the interpretation of user utterances, and with the natural language generation components to produce system utterances.
- Finally, it responds reactively to changes in its environment, including utterances from the user and other sources of information.

This section briefly elaborates on each of these aspects.

System Behavior: The CA is based on a domain-independent engine that executes tasks specified declaratively. Building on a long tradition in AI (Georgeff and Lansky 1984; Tate *et al*. 1994; Morley and Myers 2004), these tasks generally consist of goals that need to be achieved. Other tasks are invoked to achieve the sub-goals, eventually bottoming out in so-called "primitive" goals that can be achieved by built-in mechanisms. The CA execution engine includes predefined mechanisms for sequential and conditional tasks, and is easily extended.

As this is a relatively new component of the TRIPS system, the formal execution model and the semantics of tasks and operators has not yet been specified. At this point it is a programming environment for knowledge-based, goal-directed agents, not a uniform, formal representation of tasks ranging from the abstract to the highly domain-specific.

What is significant about this engine is that the agent can introspect on its execution. It can inspect the set of active and pending goals, the goal-subgoal hierarchy, the tasks chosen for active goals, and the state of those tasks. This ability is crucial for collaborative systems. In the first place, it enables intention recognition for interpreting language and identifying discourse phenomena such as topic shifts, corrections, and perhaps misunderstandings. It is also necessary to support explicit discussion of the problem-solving process, where the participants explicitly discuss what goals to pursue or whether to abandon them, for example. (The current prototype uses intention recognition against current goals for language interpretation, but not yet explicit user-initiated topic shifts. It also supports explicit abandonment of goals, but not yet richer interaction about the problem-solving process. In both cases though, we believe that the CA engine can be extended to support the more elaborate uses of introspection.)

Task Models: As noted above, the CA defines a number of abstract tasks that form the basis for the specification of behavior for a given task. Because the system is designed from the outset to support collaboration, these predefined tasks include tasks for accomplishing knowledge goals. Our representation of the system's knowledge is loosely based on standard models of knowledge and belief and their relationship to language and action (*e.g.*, Allen and Perrault 1980; Allen and Litman 1990; Cohen and Levesque 1990). Briefly, what this means is that goals may involve the system's knowing something, or more commonly that the system agree with the user about something. More generally, it could involve getting the other(s) to do something.

Standard tasks accomplish these knowledge goals by inspecting the system's beliefs regarding the content of the goal (including its beliefs about the user's beliefs) and initiating conversational acts whose effects achieve the goal. For example, to agree whether the patient is experiencing any swelling, the system does not know the answer but believes that the user does (because swelling is a type of symptom that the system knows corresponds to an internal

state of the user). As described below, this eventually results in the system asking the user. On the other hand, in agreeing about the user's weight, the system might already know the value (perhaps from an automated bathroom scale). This would result in the system informing the user as a way of reaching agreement.

For an information gathering task like the CARDIAC checkup, this framework allows the system's behavior to be specified as a set or sequence of agreement goals. These goals are the only place where domain-specific knowledge is used in the specification of behavior. This abstract, declarative specification supports intuitive user-driven behavior with effectively no additional programming.

Knowledge Base: The CA is responsible for maintaining the knowledge base (KB) that stores the system's beliefs about itself and about the user. The "state" of the interaction is a function of this knowledge and the execution engine's introspectable execution state. A few points are worth making.

We believe explicit, symbolic knowledge is crucial to support both language understanding for intuitive interaction (see below) and for realistically complex decision support. In both cases, the specific facts of the situation (patient's weight, whether they have swelling, etc.) must be combined with axiomatic (possibly probabilistic) knowledge of how these facts relate to the patient's condition and their care. The current trend towards semantically meaningful medical information systems is clearly a step in the right direction, but much work remains to be done.

The KB could be bidirectionally coupled to external sources of information, whether within TRIPS or from outside. For example, in a related project we exchanged information with a prototype PHR platform. And we have simulated the existence of connected devices whose outputs become system knowledge, such as in the scale example above.

Interaction with Natural Language Processing: The CA interacts with the natural language processing components of the system in several ways.

First, once the execution of a task has reached a conversational primitive, the CA requests that the dialogue management components perform the appropriate behavior. As noted elsewhere, this is more complex than simply turning a request to FIND-OUT-IF or FIND-OUT-WHAT into a surface question (and similarly for statements). The dialogue management components also look after requests for clarification, handling uninterpretable responses, and similar discourse phenomena without the CA's involvement.

Second, the CA supports interpretation of user utterances by a combination of intention recognition and knowledge-based interpretation. Even with topic-specific language modeling for speech recognition and deep, semantic parsing, speech recognition errors can lead to grammatical but incorrect interpretations of user input. The system uses its knowledge of what it is doing (the goal hierarchy that led to the current question, if we're interpreting an answer) and the reasoning capabilities of the knowledge base to validate interpretation hypotheses. Furthermore, for valid interpretations, the CA rewrites the linguistic interpretation into the form used by the knowledge base for the task at hand. Both of these use relatively small amounts of more or less domain-specific information, such as what sorts of things can be the location of a swelling for a CHF patient (of course, this piece of knowledge probably applies more generally than that, and so could be shared by other instantiations of the system). Crucially, the interpretation process is applied equally to the user's answers to the system's questions and to statements that the user makes on their own, supporting the user-initiated style of interaction described below.

Finally, there are complications resulting from the combination of a reactive agent (the CA) with the incremental processing of language described elsewhere. These are aggravated by the fact that the TRIPS system, in addition to embodying an assistive agent, is itself implemented as a collection of software agents. The interaction between the NL components and the CA therefore includes mechanisms for coordinating interpretation of multiple utterances or fragments, separating processing of separate contributions (possibly made without an intervening system contribution), and attempting to ensure that the agent's execution results in collaborative behavior that users find natural. This is a difficult problem in distributed computing, and an ongoing focus of work.

Reactive Behavior for Intuitive Interaction: We have emphasized that the goal of intuitive interaction requires that the system behavior be driven not by a predefined "flowchart" or "decision tree," but by an agent that can react to its environment. One example of this is the example used above of the networked scale whose output becomes part of the system's knowledge about the user. The reactive, knowledge-based, goal-driven Conversational Agent means that at any time, knowledge of the user's weight can trigger a change in behavior. If the system knows the weight before the interview, it won't ask about it (it might inform the user, depending on its beliefs about the user). If the system has asked about the weight and the user steps on the scale, the resulting knowledge would be taken as answering the question (technically, discharging the goal). These common collaborative behaviors require no programming to achieve.

More interestingly perhaps, the agent's reactive behavior and explicit goal state and task model allow it to support over-answering and, more generally, user initiative. In the case of over-answering, the content of a user's answer is taken from its linguistic interpretation, not from the fact that it is an answer to the question. The linguistic interpretation must make sense as an answer (as described above), but once committed, the entire content of the utterance becomes system knowledge. If that knowledge is the subject of subsequent goals, the CA execution engine will automatically use the knowledge and behave appropriately (for example, not asking a question for which it already knows the answer).

The system can also interpret user utterances that are not responses to questions. Again, the interpretation of the ut-

terance becomes system knowledge (and this time the interpretation must make sense relative to the task being performed, a form of intention recognition). This symmetric treatment of questions and statements means that the system automatically supports mixed-initiative interaction with no task- or domain-specific programming. Patients using our prototype for the first time typically let the system ask the questions and answer quite specifically. If they were to become more familiar with the system's goals (*i.e.*, what the system needs from them), they could make the checkup interview even simpler by describing how they're doing from the outset. The system would interpret all this, and followup only on things that we not mentioned or are not inferable from what the system knows already (from environmental sensors, PHR, *etc.*).

Evaluation

We are in the process of evaluating CARDIAC with actual chronic heart failure patients in a cardiology practice. The focus of the evaluation is whether the system can identify with high accuracy the information the patient provides in the interview. The evaluation requires the comparison of CARDIAC's analysis of patient responses with that of nurse practitioners. To this end, we created a web interface where nurse practitioners can listen to the audio of the system interviews and record their interpretation of patient responses. A detailed analysis is in progress. Preliminary observations suggest that the system can perform the CHF self-care checkup with reasonable accuracy, and that most patients believe the system is easy to use and would be helpful to them in managing their care.

Conclusion

We have described CARDIAC, an intelligent conversational assistant designed to promote successful health outcomes with patient-centered health monitoring technology. It is a first step towards developing a system that helps patients and/or their caregivers manage their medical care, provide reminders, answer questions, and engage in dialogue to collect information for monitoring a patient's current state. The information that the system collects during its interviews with the patient could be used by an interface which allowed the patient to view trends in their data, or exported to a PHR to share longitudinal data with their healthcare providers.

Acknowledgments

This research has been supported in part by NIH/NBIB grant R21HL085396 "Feasibility of Conversational Systems for Patient Care". We are grateful for helpful comments from two anonymous reviewers.

References

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, and Stent, A. 2000. An Architecture for a Generic Dialogue Shell, *Journal of Natural Language Engineering*, 6(3), pp 1-16.

Allen, J.F., and Litman, D.J. 1990. Discourse Processing and Commonsense Plans. P.R. Cohen, J. Morgan, and M. Pollack, eds., *Intentions and Communication*, MIT Press.

Allen, J. F., and Perrault, C. R. 1980. Analyzing Intention in Utterances. *Artificial Intelligence 15(3)*: 143–178.

Cohen, P., and Levesque, H. 1990. Intention is Choice with Commitment. *Artificial Intelligence 42:* 213–261.

Galescu, L., Ringger, E., and Allen, J. 1998. Rapid Language Model Development for New Task Domains. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain.

Georgeff, M. P., and Lansky, A. L. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*: 667–682.

Morley, D., and Myers, K. L. 2004. The SPARK Agent Framework. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*: 714-721. ACM Press.

Naylor, M.D., Brooten, D.A., Campbell, R.L., Maislin, G., McCauley, K.M., and Schwartz, J.S. 2004. Transitional Care of Older Adults Hospitalized with Heart Failure: A Randomized Control Trial. *Journal of American Geriatric Society, 52*: 675-684.

Tate, A., Drabble, B., and Kirby, R. 1994. O-Plan2: An Open Architecture for Command, Planning, and Control. M. Zweben and M. S. Fox, eds., *Intelligent Scheduling*: 213-240. Morgan Kaufman.