A Speech Architecture for Personal Assistants in a Knowledge Management Context

Emerson Cabrera Paraiso[1] and **Jean-Paul A. Barthès**[1] and **Cesar A. Tacla**[2]

Abstract. This paper describes the design of a speech and natural language dialog interface for Personal Assistants. We present such an architecture in a multi-agent system and apply it to knowledge management. As a clear result of this conversational speech interface, we expect an improvement in the quality of assistance.

1 INTRODUCTION

Conversational interfaces as defined by Kölzer [1] let users state what they want in their own terms, just as they would do, speaking to another person. In particular, interfacing humans to computer systems using Personal Assistants (PA) agents is a good candidate for a conversational approach. Indeed, PAs are agents that help human users (often referenced as *masters*) to do their daily work. We are convinced that a speech and conversational interface would improve the quality of assistance form a PA. We developed a spoken dialog system infrastructure for building interfaces to be used by PAs. We are applying our approach to a knowledge management (KM) multi-agent system (MAS) used in the context of research and development projects, as explained by Tacla and Barthès in [2]. The MAS has been developed to support cooperative projects, where each participant shares documents, exchanges information, and contributes to building a distributed organizational memory. To this purpose, each user is given a PA and can use plain English to control it or to ask it to perform tasks, like retrieving a document from a Lotus Notes® database or looking for knowledge in the organizational memory. The user and her PA use practical dialogs—which means that they are pursuing specific goals or tasks cooperatively as defined by Allen et al [3]. The dialog system is task-oriented. Tasks range from simple tasks like "locate a document" to more complex tasks that must be decomposed into subtasks. The nature of the application allows us to restrict the space of dialogs to those containing only Directives Speech Acts statements (e.g., inform, request, or answer). We describe now our intelligent speech architecture and how it works. We also describe briefly how the system is being used for KM and report preliminary results on the increase quality of the assistance.

2 ARCHITECTURE FOR AN INTELLIGENT SPEECH INTERFACE

The global architecture is shown in Figure 1. It has three parts: (i) graphical and speech user interface (GSUI) modules; (ii) linguistic modules; and (iii) agency modules. GSUI modules produce outputs or collect the user's inputs, like capturing voice and handling GUI events. Linguistic modules are responsible for lexical and syntactical analysis and context verification. Agency modules are directly connected to the agent kernel, that can "intelligently" manage the dialog and the interface with the help of an ontology.

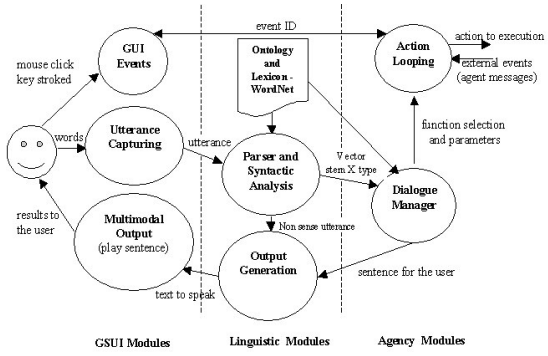

Figure 1. Interface diagram

The utterances are captured using a commercial automatic speech recognition engine that returns the recognized result for each word. The Utterance Capturing module concatenates all the words forming an utterance. A process running independently analyzes each utterance. Due to local noise interference or bad pronunciation, the utterance may be lexically and/or syntactically different from the words actually said. Initially, we are using the utterance as it is, extracting a list of known disfluencies.

The process of interpreting an utterance is done in two steps: (i) parsing and syntactic analysis; and (ii) ontology application. The results are sent to the dialog manager continuously, or back to the user if it is a nonsensical utterance. Spoken sentences have many more pronouns than written sentences. They are shorter, consisting of fragments or phrases [4]. We designed grammar rules to handle such specificities. Although our interface uses a list of specialized grammars, the latter are not restrictive.

We limited the space of dialog utterances to Directives Speech Act classes—inform, request, or answer—since they define the type of expected utterances in a master-slave relationship. The grammar rules were divided in order to classify an utterance into one of the three categories. After classification, it is possible to start the domain treatment, with the help of a domain ontology and of WordNet. Domain knowledge is used here to further process the user's statements and for reasoning. According to a taxonomy proposed by Guarino [5], they are domain and task ontologies. We are using a set of task and domain ontologies, distinguishing domain and task models for reasoning. As suggested by Allen, this

[1] Laboratoire Heudiasyc, Université de Technologie de Compiègne, BP 20529, 60.205, Compiègne, France, email: {eparaiso,barthes}@utc.fr
[2] Centro Federal de Educação Tecnologica do Parana, CEP: 80.230-901, Curitiba, PR, Brazil, email: tacla@dainf.cefetpr.br

observation is interesting for domains in which task reasoning is crucial.

Ontologies play two main roles in our PA: (i) they help interpreting the context of messages sent by others agents or by the user; and (ii) they record a computational representation of knowledge—we are using XML files—useful at inference time.

In the context of an open conversation, the problem of understanding is complex. However, one does not require a full understanding of the user's utterances to act in the right direction as stated by Popescu et al [6]. The approach to the semantic interpretation presented here is based on the notion that the meaning of utterances can be inferred by finding keywords. Precisely, the Ontology Application module is interested in finding the list of verbs that indicate the task to be executed. The corresponding keywords are concepts of the ontology directly related to a list of actions. To illustrate how this approach works, consider the utterance: *Could you list all project participants?* Since it is a question and since it is related to the application domain, the Grammar Verification module returns a matrix containing the list of tokens and their syntactic classification. By looking up the tokens in the ontology, it finds that the token *list* is an action. Note that it uses a list of synonyms from WordNet [7] (e.g. "list," "enumerate," or "name," are synonyms in this sense). It finds also that *project* is an object and *participant* is its property. At this point, we have a competence list with its parameters. Next, the Dialog Manager module takes control of the dialog. The Dialog Manager is capable of choosing a dialog model appropriate to a beginning session. For us, a dialog model contains a list of possible interactions to follow, for a given action. Each dialog session is conducted as a task with sub-tasks.

3 MAS FOR KNOWLEDGE MANAGEMENT

We are embedding our speech interface into a PA that is part of a knowledge management multi-agent system. In this architecture, agents are totally independent but belong to a cluster called a "coterie." There are two types of agents: *Service Agents* that provide a particular type of service corresponding to specific skills, and *PAs*. The assistant is in charge of all exchanges of information among participants. The PA we are working with is a rather complex system. The agent is built around three main blocks: the user interface, an *Assistancy* module and a fixed body, called the Agent Kernel. In our system all agents are cloned from a generic agent, that contains all the basic structure that allows an agent to exist, and is the kernel of each agent. The *Assistancy* module contains the mechanism for controlling the dialog and for keeping a memory of the conversation. The context of the dialog is kept by storing the competence list coming from the user interface for each dialog session that was started.

Our approach is bottom-up and aims at recording the user's behavior automatically whenever possible, building a library of cases. It comprises several steps: capturing and representing an action or operation, augmenting an operation representation, clustering operations, indexing and classifying the results. All this is done locally (by the PA and its staff) and the results are stored into a distributed memory. Actions or operations are mainly related to communications (e.g. sending emails), or documents (e.g. searching for documents). This approach presents two advantages: (i) it is easy to implement; and (ii) the information is qualified according to the needs of a particular specialist. Thus, in this approach important agents are PA rather than Service Agents. The net result is a distributed knowledge system in which the information has been organized locally as a function of the particular interests of a given specialist.

The need of a speech interface is clear when we study the complexity of a user's action in a KM system. The central point here is to decrease his cognitive overload. In addition, a KM system is a very specialized piece of software and it should not waste the user's time. In general, this kind of application involves experienced and less experienced users. Since the system reasons with user's actions, if the user details her actions, the system will produce better results. Thus, the user interface should accommodate all kinds of users, experienced or not, and provide the same results. So, our main strategy is to simplify the interaction between the user and her PA in order to reduce extra or specialized work.

4 IMPROVEMENTS IN THE QUALITY OF ASSISTANCE

Using a speech interface with a PA improves the quality of assistance in some specific situations, as in the context of KM. First of all, it makes the system operation faster and easier, since the user does not need to be an expert in KM. We hope that inexperienced users will easily operate the PA's interface compared to the ones using traditional approaches, with only GUI elements, menus and sub-menus.

Since the application is a PA, an essential feature of the user interface has been respected, namely predictability. It was an assumption we made at the beginning: to provide correct responses and act according to the user's command. Impossible requests, such as out of context, are easily handled since the system uses a competence list described as an ontology.

Although speech interfaces and dialog systems are used in several projects, our application to PAs and knowledge management is original. In addition, our contribution may also come from some design decisions. In this paper, we presented an architecture for processing conversational speech for PA in specialized domains. Such an architecture is suitable for PAs, particularly in specialized domains as KM. Our main goal is to improve the quality of the assistance. A formal evaluation process will be conducted to evaluate the results and to guide us for future improvements.

REFERENCES

[1] Kölzer, A. Universal dialogue specification for conversational systems, in *Proceedings of IJCAI 99 – Workshop on Knowledge and Reasoning in Practical Dialogue Systems.*

[2] Tacla, C.A., and Barthès, J-P. From desktop operations to lessons learned, in *Proceedings of The Seventh International Conference on CSCW* in Design, Rio de Janeiro, 2002.

[3] Allen, J. and Ferguson, G., Stent, A. An architecture for more realistic conversational systems, in *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM.

[4] Jurafsky, D. and Martin, J. *Speech and Language Processing. An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.* Prentice Hall, 2000.

[5] Guarino, N. Formal ontology in information systems, in *Proceedings of FOIS'98*, (Italy – June 1998), IOS Press, 3-15.

[6] Popescu, A.M., Etzioni, O., and Kautz, H. Towards a theory of natural language interfaces to databases, in *Proceedings of Intelligent User Interfaces 2003 (IUI-03),* ACM Press, 149-157.

[7] Fellbaum, C. *WordNet: An electronic lexical database*. MIT Press, Cambridge, MA, 1998.