Language Resources for Multi-Modal Dialogue Systems

Oliver Lemon and Alexander Gruenstein

Center for the Study of Langauge and Information
Stanford University
210 Panama Street, Stanford
CA 94305
{lemon,alexgru@csli.stanford.edu}

Abstract

This paper reviews a resource base of software agents for hub-based architectures, which can be used generally for advanced dialogue systems research and deployment. The problem of domain-specificity of dialogue managers is discussed, and we describe an approach to it developed at CSLI, involving a domain-general dialogue manager with application specific "Activity Models". We also describe relevant grammar development tools.

1. Introduction

The limited portability of dialogue-interfaces is currently a barrier to their development and more widespread use. We present and demonstrate new resources developed at CSLI[1] and elsewhere which can be used (in tandem with existing "off-the-shelf" components) to build multi-modal dialogue systems for a range of applications, each bringing their own activities and vocabulary to dialogues. We shall show that a rich resource base of software agents now exists for hub-based architectures (e.g. Galaxy Communicator (Seneff et al., 1998), OAA (Martin et al., 1999)), which can be used generally for advanced dialogue systems research and deployment.

In Section 3. we survey the components which are now available, but we first discuss common approaches and problems.

A variety of researchers have built "toolkits" for the development of dialogue systems (e.g. the OGI Toolkit (McTear, 1998), TrinidiKit (Larsson et al., 2000; Larsson and Traum, 2000)), but these have been limited to simple dialogues which are essentially pre-scripted by a developer, either as Finite State Machines or Form-filling systems (e.g. a list of questions to resolve in order to buy airline tickets). However, some current research systems handle complex non-scriptable dialogues which enable a human user to interact with a device performing co-operative activities, such as searching for objects (e.g. (Lemon et al., 2001a; Lemon et al., 2001c)). The usual problem with such systems is that their central component, the "Dialogue Manager", is application-specific, thus making them unsuitable candidates for developer toolkits for more flexible and useful dialogue systems. For example, the Pegasus, Orion, Mercury, and Jupiter systems developed at MIT (Zue et al., 1994; Seneff, 2000; Seneff and Polifroni, 2000; Zue, 2000) each have domain-specific dialogue managers, with around 350 rules each. The time and expertise needed to develop such dialogue managers by hand prohibits more widespread development and deployment of the technology.

[1]This research was (partially) funded under the Wallenberg laboratory for research on Information Technology and Autonomous Systems (WITAS) Project, Linköping University, by the Wallenberg Foundation, Sweden.

Conceptually, however, it is clear that there is a domain-independent level of conversational competence exhibited by humans. For instance, a question deserves in answer, regardless of what "domain" that question is about. Likewise, a command or instruction changes conversational context such that certain responses are responses *to* that utterance, and others are not. Capturing this level of abstraction to "speech acts" or "dialogue moves" computationally allows us to build domain-general dialogue management tools (see (Lemon et al., 2002 submitted), and Figure 1).

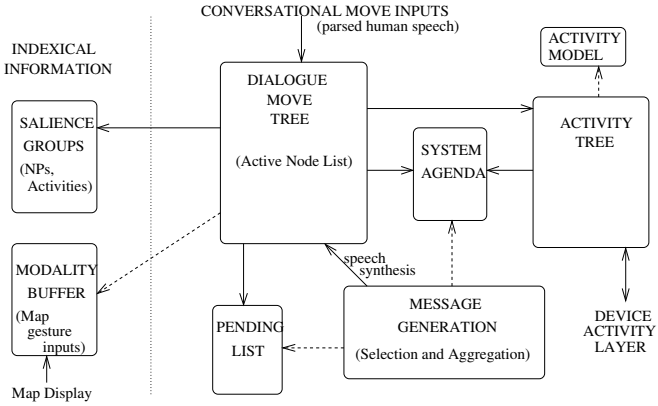

Figure 1: CSLI dialogue management system

CSLI's dialogue manager architecture and code-base enables the construction of dialogue systems for human dialogues with complex applications (e.g. the WITAS robotic helicopter, intelligent tutoring systems). The core of the system is a set of abstract Dialogue Move classes (implemented in Java), such as 'wh-question' or 'command', which contain no application-specific information, but which encode general conversational functions associated with particular speech acts (e.g. make a new NP the most salient). This resource is used in dialogue management to update conversational context, including building a "Dialogue Move Tree" representing conversational threads.

But if dialogue-management is to be domain-general, how does interaction with the underlying application take place? We have abstracted a simple language (see Section

2.) for specifying the "Activities" that an application can carry out (c.f. 'recipes' in COLLAGEN (Rich et al., 2001)), such that a developer need only specify activities, together with lexical cues for them, in order to "dialogue-enable" an application. Similar work is underway by (Rayner et al., 2002 in press). The next section outlines this idea.

2. Activity Models

The applications which we wish to "dialogue-enable" using conversational interfaces are capable of performing some basic activities or actions (possibly simultaneously). Some applications know only how to carry out sequences of atomic activities, in which case it is the dialogue system's job to decompose linguistically specified high-level activities (e.g. "record the film on channel 4 tonight") into a sequence of appropriate basic actions for the device. In this case a declarative "Activity Model" (see Figure 2) for the application states how linguistically-specified activities can be decomposed into sequences of atomic actions. This model also states traditional planning and resource constraints such as preconditions and postconditions of actions. In this way, a relatively "stupid" device (e.g. with little or no planning capabilities) can be made into a more intelligent device when it is dialogue-enabled.

We choose to focus on building this class of dialogue systems because we share with (Allen et al., 2001), a version of the the *Practical Dialogue Hypothesis*:

> "The conversational competence required for practical dialogues, although still complex, is significantly simpler to achieve than general human conversational competence."

Of course, applications may be able to plan and initiate actions themselves. Dialogue is then also used to re-specify constraints, revise activities, and monitor the progress of tasks. We propose one representation and reasoning scheme to cover the spectrum of cases from applications with no planning capabilities to those exhibiting more impressive AI. Both dialogue manager and application have access to a single "Activity Tree" which is a shared (and thus co-ordinated) representation of current and planned activities and their execution status, involving temporal and hierarchical ordering (in fact, one can think of the Activity Tree as a HTN for the application). Applications can then use the full resources of the dialogue system to report execution progress of activities, and engage the user in collaborative dialogue about them.

We now turn to a slection of available software agents for building end-to-end dialogue systems.

3. Software Agents

A collection of resources (Dialogue Manager, Activity Manager) and wrappers to "off-the-shelf" systems (Speech Recognizer, Speech Synthesizer, Parser, Generator) has been developed at CSLI as a set of portable software agents in Java, under the Open Agent Architecture (OAA2).Agents are available at our website.

The core of the architecture is OAA's "facilitator" which manages asynchronous message passing between a number

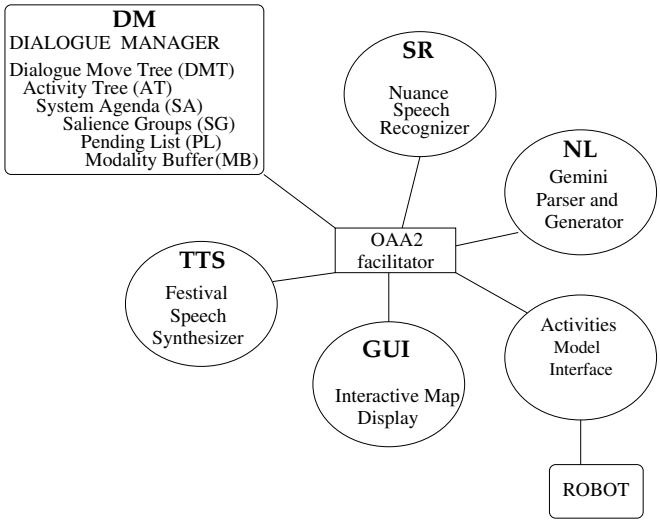

Figure 3: The WITAS multi-modal dialogue system architecture

of software agents which are specialists in certain tasks, for example speech recognition, database queries, or graphical display. In our current system there are six main agents[2] each responsible for various subtasks in the dialogue system (see Figure 3):

1. *NL (natural language):* a Prolog wrapper to SRI's "Gemini" unification-based parser and generator, using a grammar for human-robot conversation developed at CSLI.

2. *SR (speech recognizer):* a Java wrapper to a Nuance 8 speech recognition server using a language model compiled directly from the Gemini grammar (with the consequences that every recognized utterance has a logical form, and that every logical form can be mapped to a surface string).

3. *TTS (text-to-speech):* a Java wrapper to the Festival 1.4.3 speech synthesiser (Black et al., 1999), for system speech output.

4. *GUI:* an interactive map display of the current operating environment which displays route plans, waypoints, locations of vehicles including the robot, and allows deictic reference (i.e. mouse pointing) by the user.

5. *DM (dialogue manager):* co-ordinates multi-modal inputs from the user, interprets dialogue moves made by the user and application, updates and maintains the dialogue context, handles reports and questions, and sends speech and graphical outputs to the user (Java).

6. *Activity Layer:* translates commands and queries from the dialogue interface into commands and queries to the back-end application, and vice-versa for reports and queries received from the application. Uses an Activity Model (see Section 2.) and a realtime CORBA communication layer (Java, JaCORB).

[2] All are implemented in Java, but for the NL agent (Prolog).

Figure 2: A "Locate" Activity Model for a robot, exhibiting collaborative dialogue

```
task Locate  // locate is "find-by-type", collaborative activity.
            //  Breaks find into subactivities: watch_for, follow, ask_complete.
               {ResourcesUsed {camera;}       // will be checked for conflicts.
                PreConditions              //check truth of KIF statements.
                  {(Status flight inair) (Status engine ok) (Status fuel ok);}
                SkipConditions // skip this Activity if KIF condition true.
                  {(Status locked-on THIS.np);}
                PostConditions// assert these KIF statements when completed.
                  {(Status locked-on THIS.np) ;}
                Children SEQ                   //sequential sub-activities.
                  {TaskProperties
                     {command = "watch_for";   // basic robot action ---
                                 np = THIS.np;} // set sensors to search.
                   TaskProperties
                     {command = "follow_phobj";//triggers another complex activity
                                 np = THIS.np;} // following a candidate object.
                   TaskProperties              // collaborative dialogue action:
                     {command = "ask_complete";// asks user whether this is
                        np = THIS.np;  }}}     // the object we are looking for.
```

Although the GUI map and database is domain specific, the methods used in the system to determine the reference of GUI gestures (mouse clicks) are general, and can be reused in a variety of contexts[3]. Variants of some of these components have been used in other dialogue systems, notably SRI's CommandTalk (Stent et al., 1999), the NASA Personal Satellite Assistant (Rayner et al., 2000), and the robot control system of (Guzzoni et al., 1996).

As well as the components mentioned above, the following components also have wrappers developed for use in dialogue systems development under OAA (this is not a full listing, but is as comprehensive as I can be at the time of writing)[4]:

- Java Theorem Prover (Java, CSLI)
- SNARK theorem prover (Stickel et al., 2000) (Lisp, SRI)
- IBM ViaVoice speech synthesizer (C, SRI)
- Bliksem theorem prover for first-order logic, translates Discourse Representation Structures (DRS) to first-order formulae (Prolog, HCRC)
- Spass theorem prover for first-order logic, translates DRS to first-order formulae (Prolog, HCRC)
- Dipper Dialogue Move Engine (Prolog, HCRC)
- Finder, model builder for first-order logic, inputs are DRS (Prolog, HCRC)
- MACE, model-builder for first-order logic, inputs are DRS (Prolog, HCRC)
- Heyu, X-10 device control interface (HCRC)
- Left corner parser (Prolog, HCRC)
- Racer, decription logic theorem prover (Prolog, HCRC)
- TrindiKit dialogue management toolkit (Larsson and Traum, 2000) (Prolog, Gothenburg University)

The main sites for these efforts have been SRI, the NASA Rialist group, HCRC (Edinburgh), Gothenburg University (Department of Linguistics), and CSLI (Stanford University).

Various components also exist for the Galaxy Communicator hub architecture. See the Galaxy Communicator "Open Source Toolkit" at `http://communicator.sourceforge.net/download/index.html`. This includes:

- JSAPI wrapper for IBM ViaVoice or other JSAPI-compliant recognizers
- wrapper for Sphinx speech recognizer
- wrapper for Phoenix parser
- wrappers for Festival, TrueTalk speech synthesizers
- MITRE dialogue manager

3.1. Grammar Compilation

In terms of grammar development, there are also a number of compilers, for converting Unification Grammars (e.g. developed in Gemini) into context free grammars which can be used to build language models for Nuance speech recognition. For example:

[3] e.g. any application where items are selected by point-and-click gestures. Dragging and dropping would be nice to add.

[4] Where possible I note in parentheses the language used and the institution at which the work is based.

- Regulus (Rayner et al., 2001)
- Kiefer and Kreiger (Kiefer and Krieger, 2000)
- Gemini (Dowding et al., 1993)
- UNIANCE[5]

Papers investigating this issue are (Moore, 1998; Dowding et al., 2001; Rayner et al., 2001). The virtue of this process is that developers need only write one unification grammar for the application, which can be used in three ways: interpretation, generation (if the system is bi-directional), and speech recognition.

Another challenge is that multiple application-specific grammars are often used for parsing, generation, and speech recognition. Recent developments in "plug-and-play" grammars (Rayner et al., 2002 in press) allow applications and devices to bring their own specific vocabulary and grammar rules to a domain general "core" grammar, and use them on the fly.

4. Conclusion

We described a number of software resources available for advanced dialogue systems research and deployment. These include hub architectures, parsers, speech recognizers, speech synthesizers, dialogue managers, and toolkits, as well as development tools such as software for compiling unification grammars to context-free grammars for speech recognition.

At CSLI, these tools have been used to rapidly prototype dialogue systems for interaction with intelligent autonomous devices such as mobile robots (Lemon et al., 2001c) and intelligent tutoring systems (Clark et al., 2001).

We also discussed the problem of domain-specificity of dialogue systems – a problem which is especially acute in the core area of dialogue management. We outlined the approach taken to this at CSLI, based on the representation of domain-specific "activity models" for devices, applications, or services. This allows dialogue management to be handled in a general fashion, in our case using abstract "dialogue move" classes to build and update a "dialogue move tree" representing dynamic dialogue context.

More information and demos see `http://www-csli.stanford.edu/semlab/witas/`
Other applications include interaction with avatars and computer-generated characters in video games (Lemon, 2002).

Acknowledgements

With thanks to John Dowding (NASA Rialist), Beth-Ann Hockey (NASA Rialist), Stina Ericsson (Gothenburg), Johan Bos (HCRC), Staffan Larsson (Gothenburg), Stanley Peters (CSLI).

[5]See `http://www.iccs.informatics.ed.ac.uk/~jbos/systems.html`

5. References

James F. Allen, Bradford W. Miller, Eric K. Ringger, and Teresa Sikorski. 1996. A robust system for natural spoken dialogue. In *Proceedings of ACL*.

James Allen, Donna Byron, Myroslva Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. 2001. Toward conversational human-computer interaction. *AI Magazine*, 22(4):27–37.

Alan Black, Paul Taylor, and Richard Caley. 1999. The Festival Speech Synthesis system. `http://www.cstr.ed.ac.uk/projects/festival/`.

Peter Bohlin, Robin Cooper, Elisabet Engdahl, and Staffan Larsson. 1999. Information states and dialog move engines. *Electronic Transactions in AI*. Website with commentaries: `www.etaij.org`.

Brady Clark, John Fry, Matt Ginzton, Stanley Peters, Heather Pon-Barry, and Zachary Thomsen-Gray. 2001. Automated tutoring dialogues for training in shipboard damage control. In *Proceedings of SIGDIAL 2001*.

Robin Cooper and Staffan Larsson. 1998. Dialog moves and information states. Technical Report 98-6, Goteborg University. Gothenburg papers in Computational Linguistics.

John Dowding, Jean Mark Gawron, Doug Appelt, John Bear, Lynn Cherny, Robert Moore, and Douglas Moran. 1993. GEMINI: a natural language system for spoken-language understanding. In *Proc. 31st Annual Meeting of the ACL*.

J. Dowding, B.A. Hockey, M. J. Gawron, and C. Culy. 2001. Practical issues in compiling typed unification grammars for speech recognition. In *Proceedings of the Thirty-Ninth Annual Meeting of the Association for Compuational Linguistics*.

Renee Elio and Afsaneh Haddadi. 1999. On abstract task models and conversation policies. In *Workshop on Specifying and Implementing Conversation Policies, Autonomous Agents'99*, Seattle.

John Fry, Hideki Asoh, and Toshihiro Matsui. 1998. Natural dialogue with the Jijo-2 office robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS-98*, pages 1278–1283, Victoria, B.C., Canada. (See `www-csli.stanford.edu/semlab/juno`).

Dafydd Gibbon, Inge Mertins, and Roger Moore. 2000. *Handbook of Spoken and Multi-modal Dialogue Systems*. Kluwer.

Didier Guzzoni, Adam Cheyer, Luc Julia, and Kurt Konolige. 1996. Many robots make short work. In *AAAI Robotics Contest*, Menlo Park, CA. SRI International, AAAI Press.

B. Kiefer and H. Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of 6th International Workshop on Parsing Technologies*, pages 135–146.

Staffan Larsson and David Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340.

S. Larsson, P. Bohlin, J. Bos, and D. Traum. 2000.

TRINDIKIT 1.0 Manual. Technical report, University of Gothenburg.

Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. 2001. Training personal robots using natural language instruction. *IEEE Intelligent Systems*.

Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. 2001a. Information states in a multi-modal dialogue system for human-robot conversation. In Peter Kühnlein, Hans Reiser, and Henk Zeevat, editors, *5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog 2001)*, pages 57 – 67.

Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. 2001b. A multi-modal dialogue system for human-robot conversation. In *Proceedings of North American Association for Computational Linguistics (NAACL 2001)*.

Oliver Lemon, Anne Bracy, Alexander Gruenstein, and Stanley Peters. 2001c. The WITAS Multi-Modal Dialogue System I. In *Proceedings of 7th European Conference on Speech Communication and Technology (Eurospeech' 01)*, Aalborg.

Oliver Lemon, Alexander Gruenstein, and Stanley Peters. 2002 (submitted). Collaborative actvities and multi-tasking in dialogue systems. *Traitement Automatique des Langues (TAL)*. Special Issue on Dialogue.

Oliver Lemon. 2002. Transferable multi-modal dialogue systems for interactive entertainment. In *AAAI Spring Symposium on Artificial Intelligence in Interactive Entertainment*, Technical Report SS-02-01, pages 57 – 61, Menlo Park, CA. AAAI Press.

Diane Litman, Micheal Kearns, Satinder Singh, and Marilyn Walker. 2000. Automatic optimization of dialogue management. In *Proceedings of COLING 2000*.

Susann LuperFoy, Dan Loehr, David Duff, Keith Miller, Florence Reeder, and Lisa Harper. 1998. An architecture for dialogue management, context tracking, and pragmatic adaptation in spoken dialogue systems. In *COLING-ACL*, pages 794 – 801.

David Martin, Adam Cheyer, and Douglas Moran. 1999. The Open Agent Architecture: a framework for building distributed software systems. *Applied Artificial Intelligence: An International Journal*, 13(1-2).

M. McTear. 1998. Modelling spoken dialogues with state transition diagrams: Experiences with the CSLU toolkit. In *Proc 5th International Conference on Spoken Language Processing*.

R. Moore. 1998. Using natural language knowledge sources in speech recognition. In *Proceedings of the NATO Advanced Studies Institute*.

Douglas Moran, Adam Cheyer, Luc Julia, David Martin, and Sangkyu Park. 1997. Multimodal user interfaces in the Open Agent Architecture. In *Proc IUI 97*, pages 61 – 68.

Nuance. 2000. `http://www.nuance.com`.

James Pittman, Ira Smith, Phil Cohen, Sharon Oviatt, and Tzu-Chieh Yang. 1996. Quickset: a multimodal interface for military simulation. In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, Orlando*, pages 217–224.

Manny Rayner, Beth Ann Hockey, and Frankie James. 2000. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of Applied Natural Language Processing (ANLP)*.

M. Rayner, J. Dowding, and B.A. Hockey. 2001. A baseline method for compiling typed unification grammars into context free language models. In *Proceedings of EuroSpeech*.

Manny Rayner, John Boye, Ian Lewin, and Genevieve Gorrel. 2002 (in press). Plug and play spoken dialogue processing. In *SIGdial 2001 book*. Kluwer.

Charles Rich, Candace Sidner, and Neal Lesh. 2001. Collagen: applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–25.

Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialog management for robots. In *Proceedings of ACL 2000*.

S. Seneff and J. Polifroni. 2000. Formal and natural language generation in the mercury conversational system.

S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue. 1998. Galaxy-ii: A reference architecture for conversational system development. In *Proceedings of ICSLP*.

Stephanie Seneff. 2000. Orion: From on-line interaction to off-line delegation. In *Proc. 6th International Conference on Spoken Language Processing*.

Amanda Stent, John Dowding, Jean Mark Gawron, Elizabeth Owen Bratt, and Robert Moore. 1999. The CommandTalk spoken dialogue system. In *Proceedings of the Thirty-Seventh Annual Meeting of the ACL*, pages 183–190, University of Maryland, College Park, MD. Association for Computational Linguistics.

Mark E. Stickel, Richard J. Waldinger, and Vinay K. Chaudhri. 2000. A Guide to SNARK. Technical Note Unassigned, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, May.

Wei Xu and Alexander Rudnicky. 2000. Task-based dialog management using an agenda. In *Proceedings of ANLP/NAACL 2000 Workshop on Conversational Systems*, pages 42–47.

V. Zue, S. Seneff, J. Polifroni, M. Phillips, C. Pao, D. Goddeau, J. Glass, E., and Brill. 1994. PEGASUS: A spoken language interface for on-line air travel planning. In *Proc. ARPA Human Language Technology Workshop '94*, Princeton, NJ.

V. Zue. 2000. Jupiter: A telephone-based conversational interface for weather information.