Multi-Human Dialogue Understanding for Assisting Artifact-Producing Meetings

John Niekrasz and **Alexander Gruenstein** and **Lawrence Cavedon**
Center for the Study of Language and Information (CSLI)
Stanford University
Cordura Hall, Stanford, CA, 94305-4115, USA
http://www-csli.stanford.edu/semlab/
{niekrasz, alexgru, lcavedon}@csli.stanford.edu

Abstract

In this paper we present the dialogue-understanding components of an architecture for assisting multi-human conversations in artifact-producing meetings: meetings in which tangible products such as project planning charts are created. Novel aspects of our system include multimodal ambiguity resolution, modular ontology-driven artifact manipulation, and a meeting browser for use during and after meetings. We describe the software architecture and demonstrate the system using an example multimodal dialogue.

1 Introduction

Recently, much attention has been focused on the domain of multi-person meeting understanding. Meeting dialogue presents a wide range of challenges including continuous multi-speaker automatic speech recognition (ASR), 2D whiteboard gesture and handwriting recognition, 3D body and eye tracking, and multimodal multi-human dialogue management and understanding. A significant amount of research has gone toward understanding the problems facing the collection, organization, and visualization of meeting data (Moore, 2002; Waibel et al., 2001), and meeting corpora like the ICSI Meeting Corpus (Janin et al., 2003) are being made available. Continuing research in the multimodal meeting domain has since blossomed, including ongoing work from projects such as AMI[1] and M4[2], and efforts from several institutions.

Previous work on automatic meeting understanding has mostly focused on surface-level recognition, such as speech segmentation, for obvious reasons: understanding free multi-human speech at any level is an extremely difficult problem for which best performance is currently poor. In addition, the primary focus for applications has been on off-line tools such as post-meeting multimodal information browsing.

In parallel to such efforts we are applying dialogue-management techniques to attempt to understand and monitor meeting dialogues as they occur, and to supplement multimodal meeting records with information relating to the structure and purpose of the meeting.

Our efforts are focused on assisting *artifact-producing* meetings, i.e. meetings for which the intended outcome is a tangible product such as a project management plan or a budget. The dialogue-understanding system helps to create and manipulate the artifact, delivering a final product at the end of the meeting, while the state of the artifact is used as part of the *dialogue context* under which interpretation of future utterances is performed, serving a number of useful roles in the dialogue-understanding process:

- The dialogue manager employs generic dialogue moves with *plugin points* to be defined by specific artifact types, e.g. project plan, budget;
- The artifact state helps *resolve ambiguity* by providing evidence for multimodal fusion and constraining topic-recognition;
- The artifact type can be used to *bias ASR language-models*;
- The constructed artifact provides a interface for a *meeting browser* that supports directed queries about discussion that took place in the meeting, e.g. "Why did we decide on that date?"

In addition, we focus our attention on the handling of ambiguities produced on many levels, including those produced during automatic speech recognition, multimodal communication, and artifact manipulation. The present dialogue manager uses several techniques to do this, including the maintenance of

[1] http://www.amiproject.org/
[2] http://www.m4project.org/

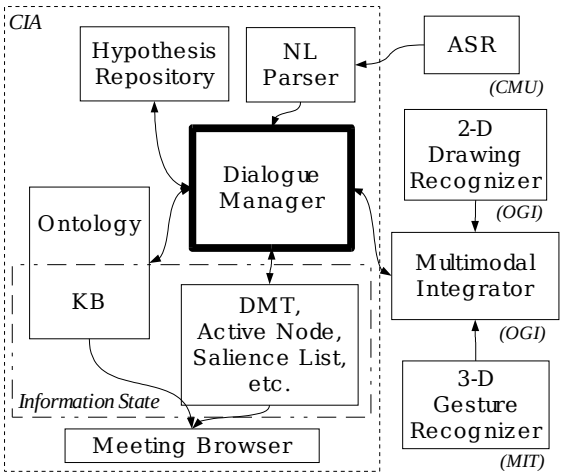


Figure 1: The meeting assistant architecture, highlighting the dialogue-management components.

multiple dialogue-move hypotheses, fusion with multimodal gestures, and the incorporation of artifact-specific plug-ins.

The software architecture we use for managing multi-human dialogue is an enhancement of a dialogue-management toolkit previously used at CSLI in a range of applications, including command-and-control of autonomous systems (Lemon et al., 2002) and intelligent tutoring (Clark et al., 2002). In this paper, we detail the dialogue-management components (Section 3), which support a larger project involving multiple collaborating institutions (Section 2) to build a multimodal meeting-understanding system capable of integrating speech, drawing and writing on a whiteboard, and physical gesture recognition.

We also describe our toolkit for on-line and off-line meeting browsing (Section 4), which allows a meeting participant, observer, or developer to visually and interactively answer questions about the history of a meeting, the processes performed to understand it, and the causal relationships between dialogue and artifact manipulation.

2 Meeting Assistant Architecture

The complete meeting assistant architecture is a highly collaborative effort from several institutions. Its overall architecture, focusing on our contributions to the system is illustrated in Figure 1.

The components for drawing and writing recognition and multimodal integration (Kaiser et al., 2003) were developed at The Oregon Graduate Institute (OGI) Center for Human-Computer Communication[3]; the component for physical gesture recognition (Ko et al., 2003) was developed at The Massachusetts Institute of Technology (MIT) AI Lab[4]. Integration between all components was performed by project members at those sites and at SRI International[5], and integration between our CSLI Conversational Intelligence Architecture and OGI's Multimodal Integrator (MI) was performed by members of both teams. ASR is done using CMU Sphinx[6], from which the n-best list of results are passed to SRI's Gemini parser (Dowding et al., 1993). Gemini incorporates a suite of techniques for handling noisy input, including fragment detection, and its dynamic grammar capabilities are used to register new lexical items, such as names of tasks that may be out-of-grammar.

An example of a multimodal meeting conversation that the meeting assistant currently supports can be found in Figure 2.[7] There are two meeting participants in a conference room with an electronic whiteboard which can record their pen strokes and a video camera that tracks their body movements; A is standing at the whiteboard and drawing while B is sitting at the table. A gloss of how the system behaves in response to each utterance and gesture follows each utterance; these glosses will be explained in greater detail throughout the rest of the paper. The drawing made on the whiteboard is in Figure 3(a), and the chart artifact as it was constructed by the system is displayed in Figure 3(b).

3 Conversational Intelligence Architecture

To meet the challenges presented by multi-person meeting dialogue, we have extended and enhanced our previously used Conversational Intelligence Architecture (CIA). The CIA is a modular and highly configurable multi-application system: a separation is made between generic dialogue processes and those specific to a particular domain. Creating a new application may involve writing new dialogue moves and configuring the CIA to use these. We

[3]http://www.cse.ogi.edu/CHCC/
[4]http://www.ai.mit.edu/
[5]http://www.sri.com/
[6]http://www.speech.cs.cmu.edu/sphinx/
[7]A video demonstration will be available soon at http://www-csli.stanford.edu/semlab/calo/

A: So, lets uh figure out what uh needs uh needs to be done. Let's look at the schedule. [draws a chart axes] *utterance and gesture information fused, a new milestone chart artifact is created*

B: So, if all goes well, we've got funding for five years. *system sets unit on axis to "years"*

A: Yeah. Let's see one, two ... [draws five tick marks on the x-axis] *system assumes tick marks are years*

B: Well, the way I see it, uh we've got three tasks. *dialogue manager hypothesizes three tasks should be added, waits for multimodal confirmation*

A: Yeah right [draws three task lines horizontally on the axis] *multimodal confirmation is given, information about task start and end dates is fused from the drawing*

A: Let's call this task line demo [touches the top line with the pen], call this task line signoff [touches the middle line with the pen], and call this task line system [touches the bottom line with the pen]. *each utterance causes the dialogue manager to hypothesize three distinct hypotheses, in each task a different hypothesis is named, the gestures disambiguate these in the multimodal integrator*

B: So we have two demos to get done.

A: uh huh

B: Darpatech is at the end of month fifteen [*A* draws a diamond at month fifteen on the demo task line] *dialogue manager hypothesizes a milestone called "darpatech" at month fifteen; gesture confirms this and pinpoints appropriate task line*

B: And the final demonstrations are at the end of year five [*A* draws a diamond at year five on the demo task line] *same processing as previous*

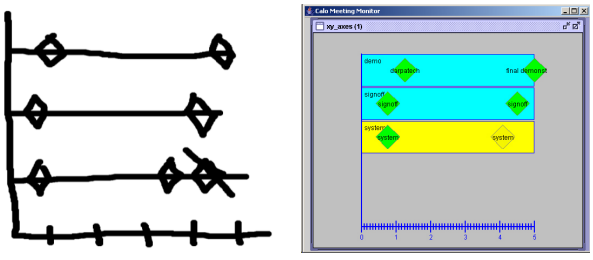
A: Hmm, so when do the signoffs need to happen do you think? *dialogue manager expects next utterance to be an answer*

B: Six months before the demos [*A* draws two diamonds on the signoff task line, each one about 6 months before the demo milestones drawn above] *answer arrives; dialogue manager hypothesizes two new milestones which are confirmed by gesture*

A: And we'll need the systems by then too [*A* draws two diamonds on the system task line] *dialogue manager hypothesizes two more milestones, confirmed by gesture*

B: That's a bit aggressive I think. Let's move the system milestone back six months. [*B* points finger at rightmost system milestone. *A* crosses it out and draws another one six months earlier] *dialogue manager hypothesizes a move of the milestone, 3D gesture and drawing confirm this*

Figure 2: Example conversation understood by the system.



(a) The whiteboard input captured by OGI's Charter gesture recognizer

(b) The artifact as maintained in the dialogue system

Figure 3: Ink-captured vs 'idealized' artifact output.

have successfully used this "toolkit" approach in our previous applications at CSLI to interface novel devices without modifying the core dialogue manager.

The present application is however very different to our previous applications, and those commonly encountered in the literature, which typically involve a single human user interacting with a dialogue-enabled artificial agent. In the meeting environment, the dialogue manager should at most very rarely interpose itself into the discussion—to do so would be disruptive to the interaction between the human participants. This requirement prohibits ambiguity and uncertainty from being resolved with, say, a clarification question, which is the usual strategy in conversational interfaces. Instead, uncertainty must be maintained in the system until it can be resolved by leveraging context, using evidence from another modality, or by a future utterance.

The meeting-understanding domain has thus prompted several extensions to our existing CIA, many of which we expect will be applied in other conversational domains. These include:

- Support for handling multiple competing speech parses; (Section 3.2)
- A generic artifact ontology which enables designing generically useful artifact-savvy dialogue applications; (Section 3.3)
- Support for the generation and subsequent confirmation of dialogue-move hypotheses in a multimodal integration framework; (Section 3.4)
- The acceptance of non-verbal unimodal gestures into the dialogue-move repertoire. (Section 3.5)
- A preliminary mechanism for supporting uncertainty across multiple conversational moves; (Section 3.6)

Before discussing these new features in detail, the following section introduces the CIA and its persisting core dialogue-management components.

3.1 Core Components: Information State and Context

The core dialogue management components of the CIA maintain dialogue context using the *information-state* and *dialogue-move* approach (Larsson and Traum, 2000) where each contributed utterance modifies the current context, or information state, of the dialogue. Each new utterance is then interpreted within the current context (see (Lemon et al., 2002) for a detailed description).

A number of data structures are employed in this process. The central dialogue state-maintaining structure is the *Dialogue Move Tree (DMT)*. The DMT represents the historical context of a dialogue. An incoming utterance, classified by dialogue move, is interpreted in context by attaching itself to an appropriate active node on the DMT; e.g., an answer attaches to an active corresponding question node. Currently, active nodes are kept on an *Active Node List*, which is ordered so that those most likely to be relevant to the current conversation are at the front of the list. Incoming utterances are displayed to each node in turn, and attach to the first appropriate node (determined by information-state-update functions). Other structures include the context-specific *Salience List*, which maintains recently used terms for performing anaphora resolution, and a *Knowledge Base* containing application specific information, which may be leveraged to interpret incoming utterances.[8]

We now present the various enhancements made to the CIA for use in the meeting domain.

3.2 ASR and Robust Parsing

The first step in understanding any dialogue is recognizing and interpreting spoken utterances. In the meeting domain, we are presented with the particularly difficult task of doing this for spontaneous human-human speech. We therefore chose to perform ASR using a statistical language model (LM) and employ CMU's Sphinx to generate an n-best list of recognition results. The recognition engine uses a trigram LM trained on the complete set of possible utterances expected given a small handcrafted scenario like that in the example dialogue. Despite the task's limited domain, the realized speech is very disfluent, generating an extremely broad range of possible utterances that the system must handle. The resulting n-best list is therefore often extremely varied.

To handle the ASR results of disfluent utterances, we employ SRI's Gemini robust language parser (Dowding et al., 1993). In particular, we use Gemini to retrieve the longest strings of valid *S* and *NP* fragments in each ASR result. Currently, we reject all but the parsed *S* fragments—and *NP* fragments when expected by the system (*e.g.* an answer to a question containing an *NP* gap). The parser uses generic syntactic rules, but is constrained semantically by *sorts* specific to the domain. In Section 3.4, we describe how the dialogue manager handles the multiple parses for a single utterance and how it uses the uncertainty they represent.

3.3 Artifact Knowledge Base and Ontology

In the present version of the CIA, all static domain knowledge about meeting artifacts is defined in a modularized class-based ontology. In conjunction with the ontology, we also maintain a dynamic knowledge base (KB) which holds the current state of any artifacts. This is stored as a collection of instances of the ontological classes, and both components are maintained together using the Protégé-2000[9] ontology and knowledge-base toolkit (Grosso et al., 1999).

The principal base classes in the artifact ontology are designed to be both architecturally elegant and intuitive. To this end, we characterize the world of artifacts as being made up of three essential classes: *entities* which represent the tangible objects themselves, *relations* which represent how the entities relate to one another, and *events* which change the state of entities or relations. Events are the most important tool aiding the dialogue management algorithm. They comprehensively characterize the set of actions which can change the current state of an artifact. They may be classified into three categories: *insert changes* which insert a new entity or relation instance into the KB, *remove changes* which remove an instance, and *value changes* which modify the value of a slot in an instance. All changes to the KB can be characterized as one of these three atomic events or a combination of them.

3.4 *Hypothesizers*: A plugin architecture for artifact-driven multimodal integration

Abmiguities and uncertainties are both rampant in multimodal meeting dialogues, and in artifact-producing meetings, the majority pertain to artifacts and the utterances performed to change them. In this section we explain how the CIA's dialogue manager uses the artifact ontology, and the repertoire of event classes in it, to formulate sets of artifact-changing dialogue-move hypotheses from single utterances. We

[8] Command-and-control applications have also made use of an *Activity Tree*, which represents activities being carried out by the dialogue-enabled device (Gruenstein, 2002); however, this application currently makes no use of this.

[9] http://protege.stanford.edu/

also demonstrate how it uses the current state of the artifact in the KB to constrain the interpretation of utterances in context, and how multimodal gestures help to resolve ambiguous interpretations.

To begin, each dialogue-move hypothesis consists of the following elements: (1) the *DMT node* associated with this hypothesis, (2) the *parse* that gave rise to the hypothesis, (3) the *probability* of the hypothesis, (4) an *isUnimodal* flag indicating whether or not the dialogue move requires confirmation from other modalities, (5) a list of *artifact-change events* to be made to the KB, and (6) the *information state update function* to be invoked if this hypothesis is confirmed by the multimodal integrator. Each of these elements participate in the generation and confirmation process as detailed below.

First, consider the utterance *Darpatech is at the end of month fifteen.* from the example dialogue. This utterance is much more likely to indicate the creation of a new milestone if a task line is pertinent to the current dialogue context, e.g. the user has just created a new task line. In our system, the ambiguous or uncertain utterance, the current dialogue context, and the current state of the chart is delegated to artifact-type specific components called *hypothesizers*. Hypothesizers take the above as input, and using the set of events available to its corresponding artifact in the ontology, they programmatically generate a list of dialogue-move hypotheses appropriate in the given context—or they can return the empty list to indicate that there is no reasonable interpretation of the utterance given the current context.

Hypothesizers work directly with the DMT architecture: as an incoming utterance is sequentially presented to each active node in the DMT, the dialogue context and the proposed active node are passed into a hypothesizer corresponding to the particular artifact associated with that node. If the hypothesizer can create one or more valid hypotheses, then the utterance is attached to the DMT as a child of that active node.[10]

In a multimodal domain, some hypotheses require confirmation in other modalities before the dialogue manager can confidently update the information state. In this particular system, in fact, the dialogue manager does not directly update the KB's current artifact state; rather, it hypothesizes a set of dialogue-move hypotheses and assigns each a confidence derived from ASR confidence, the fragmentedness of the parse, and confidence in the proposed attachment to a conversational thread. Each conversational move is then provided a *Hypothesis Repository* for storing the hypotheses associated with it. When dialogue processing is completed for a particular conversational move, *i.e.* when all possible attachments of all possible parses on the n-best list have been made, the set of hypotheses is sent to the Multimodal Integrator (MI) for potential fusion with gesture. Depending on the information from other modalities, the MI confirms or rejects the hypotheses—moreover, a confirmed hypothesis might be *augmented* with information provided by other modalities. Such an augmentation occurs for the utterance *We have three tasks* from the example dialogue. In this situation, the dialogue manager hypothesizes that the user may be creating three new task lines on the chart. When the user actually draws the three task lines, the MI infers the start and stop date based on where the lines start and stop on the axis. In this case, it not only confirms the dialogue manager's hypothesis, but augments it to reflect the additional date information yielded from the whiteboard input.

3.5 Unimodal Gestures

In addition to the Information State updates based on both speech and gesture, multimodal meeting dialogue can often include gestures in which a participant makes a change to an artifact using a *unimodal* gesture not associated with an utterance. For example, a user may draw a diamond on a task line but say nothing. Even in the absence of speech, this can be unambiguously understood as the creation of a milestone at a particular point on the line. These unimodally produced changes to the chart must be noted by the dialogue manager, as they are potential targets for later conversation. To accommodate this, we introduce a new DMT node of type *Unimodal Gesture*, thus implicitly including gesture as a communicative act that can stand on its own in a conversation

3.6 Uncertain DMT Node Attachment

Since hypotheses are not always immediately confirmed, uncertainty must be maintained

[10] There are, in fact, other rules as well which allow for attachment. For example, questions—which don't immediately generate hypotheses—can also be attached to various nodes depending on the dialogue context. While the emphasis here is on hypothesizers, these are just one part of the dialogue processing toolkit

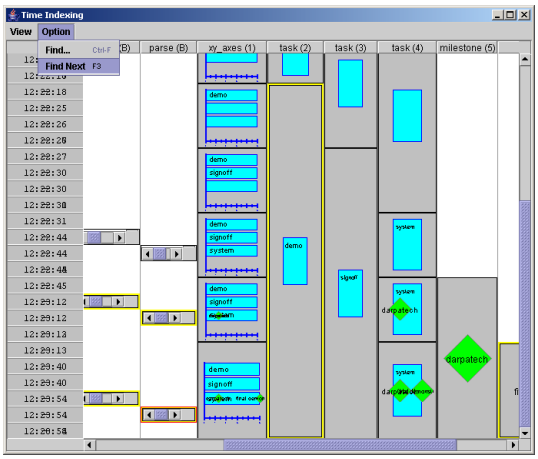


Figure 4: A snapshot from the meeting browser.

across multiple dialogue moves. The system accomplishes this by extending the CIA to maintain multiple competing *Information States*. In particular, the DMT has been extended to allow for the same parse to attach in multiple locations—these multiple attachments are eventually pruned as more evidence is accumulated in the form of further speech or gestures—that is, as hypotheses are confirmed or rejected over time.

4 Meeting Viewer Toolkit

Throughout an artifact-producing meeting, the dialogue system processes a complex chronological sequence of events and information states that form structures rich in information useful to dialogue researchers and the dialogue participants themselves. To harness the power of this information, we have constructed a toolkit for visualizing and investigating the meeting information state and its history.

Central to the toolkit is our *meeting history browser*, which can be seen in Figure 4, displaying a portion of the example dialogue, with the results of a search for "demo" highlighted. This record of the meeting is available both during the meeting and afterwards to assist users in answering questions they might have about the meeting. Many kinds of questions can be answered in the browser, like those a manager might ask the day after a meeting: *"Why did we move the deadline on that task 6 months later?"*, *"Did I approve setting that deadline so early?"*, and *"What were we thinking when we put that milestone at month fifteen?"*. A meeting participant might have questions as the meeting occurs, like *"What did the chart look like 5 minutes ago?"*, *"What did we say to make the system move that milestone?"*, and *"What did Mr. Smith say at the beginning of the meeting?"*.

To help answer these questions, the browser performs many of the functions found in current multimodal meeting browsers. For example, it provides concise display of a meeting transcription, advanced searching capabilities, saving and loading of meeting sessions, and personalization of its own display characteristics. As a novel addition to these basic behaviors, the browser is also designed to display artifacts and the causal relationships between artifacts and the utterances that cause them to change.

To effectively convey this information, the record of components monitored by the history toolkit is presented to the user through a window which chronologically displays the visual embodiment of those components. Recognized utterances are shown as text, parses are shown as grouped string fragments, and artifacts and their sub-components are shown in their prototypical graphical form. The window organizes these visual representations of the meeting's events and states into chronological tracks, each of which monitors a unified conceptual part of the meeting. The user is then able to link the elements causally.

Beyond the history browser, the toolkit also displays the current state of all artifacts in an *artifact-state window* (e.g. Figure 3(b)). In the window, the user not only confirms the state of the artifact but can also gain insight into the currently interpreted dialogue context by monitoring how the artifact is *highlighted*. In the figure, the third task is highlighted because it is the most recently talked-about task. A meeting participant can therefore see that subsequent anaphoric references to an unknown task will be resolved to the third one.

Another GUI component of the toolkit is a small *hypothesis window* which shows the current set of unresolved artifact-changing hypotheses. It does this by displaying an artifact for each hypothesis, reflecting the artifact's future state given confirmation of the hypothesis. The hypothesis' probability and associated parse is displayed under the artifact. The user may even directly click a hypothesis to confirm it. The hypothesized future states are however not displayed in the artifact-state window or artifact-history browser, which show only the results of confirmed actions.

In addition to being a GUI front-end, the toolkit maintains a fully generic architecture for

recording the history of any object in the system software. These objects can be anything from the utterances of a participant, to the state history of an artifact component, or the record of hypotheses formulated by the dialogue manager. This generic functionality provides the toolkit the ability to answer a wide variety of questions for the user about absolutely any aspect of the dialogue context history.

5 Future Work

Work is currently proceeding in a number of directions. Firstly, we plan to incorporate further techniques for robust language understanding, including word-spotting and other topic-recognition techniques, within the context of the constructed artifact. We also plan to investigate using the current state of the artifact to further bias the ASR language model. We also plan on generalizing the uncertainty management within the dialogue manager, allowing multiple competing hypotheses to be supported over multiple dialogue moves. Topic and other ambiguity management techniques will be used to statistically filter and bias hypotheses, based on artifact state.

We are currently expanding the meeting browser to categorize utterances by dialogue act, and to recognize and categorize aggregations as multi-move strategies, such as *negotiations.* This will allow at-a-glance detection of where disagreements took place, and where issues may have been left unresolved. A longer-term aim of the project is to provide further support to the participants in the meeting, e.g. by detecting opportunities to provide useful information (e.g. schedules, when discussing who to allocate to a task; documents pertinent to a topic under discussion) to meeting participants automatically. Evaluation criteria are currently being designed that include both standard measures, such as word error rate, and measures involving recognition of meeting-level phenomena, such as detecting agreement on action-items. Evaluation will be performed using both corpus-based approaches (e.g. for evaluating recognition of meeting phenomena) and real (controlled) meetings with human subjects.

6 Acknowledgements

We would like to gratefully acknowledge Phil Cohen's group at OGI, especially Ed Kaiser, Xiaoguang Li, and Matt Wesson, and David Demirdjian at MIT. This work was funded by DARPA grant NBCH-D-03-0010(1).

References

B. Clark, E. Owen Bratt, O. Lemon, S. Peters, H. Pon-Barry, Z. Thomsen-Gray, and P. Treeratpituk. 2002. A general purpose architecture for intelligent tutoring systems. In *International CLASS Workshop on Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems.*

J. Dowding, J.M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: a natural language system for spoken-language understanding. In *Proc. ACL 93.*

W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, and M. A. Musen. 1999. Knowledge modeling at the millennium: (the design and evolution of Protégé-2000). In *Proc. KAW 99.*

A. Gruenstein. 2002. Conversational interfaces: A domain-independent architecture for task-oriented dialogues. Master's thesis, Stanford University.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The ICSI meeting corpus. In *Proc. ICASSP 2003.*

E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, P. Cohen, and S. Feiner. 2003. Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *Proc. ICMI 2003.*

T. Ko, D. Demirdjian, and T. Darrell. 2003. Untethered gesture acquisition and recognition for a multimodal conversational system. In *Proc. ICMI 2003.*

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6.

O. Lemon, A. Gruenstein, and S. Peters. 2002. Collaborative activities and multi-tasking in dialogue systems. *Traitment Automatique des Langues*, 43(2).

D. Moore. 2002. The IDIAP smart meeting room. Technical Report IDIAP Communication 02-07.

A. Waibel, M. Bett, F. Metze, K. Ries, T. Schaaf, T. Schultz, H. Soltau, H. Yu, and K. Zechner. 2001. Advances in automatic meeting record creation and access. In *Proc. ICASSP 2001.*