Combining Confidence Scores with Contextual Features for Robust Multi-Device Dialogue*

Lawrence Cavedon
National ICT Australia, Victoria Research Lab
and CS&IT, RMIT University
Melbourne VIC, Australia
lawrence.cavedon@nicta.com.au

Matthew Purver, Florin Ratiu
CSLI, Stanford University
Cordura Hall, 210 Panama St. Stanford
CA 94305, USA
{mpurver,fratiu}@stanford.edu

Abstract

We present an approach to multi-device dialogue that evaluates and selects amongst candidate dialogue moves based on features at multiple levels. Multiple sources of information can be combined, multiple speech recognition and parsing hypotheses tested, and multiple devices and moves considered to choose the highest scoring hypothesis overall. The approach has the added benefit of potentially re-ordering n-best lists of inputs, effectively correcting errors in speech recognition or parsing. A current application includes conversational interaction with a collection of in-car devices.

1 Introduction

In this paper, we describe recent enhancements to the *CSLI Dialogue Manager* (CDM) infrastructure to increase robustness, in particular in (but not exclusive to) multi-device settings. Dialogue contributions may be processed using multiple information sources (e.g. deep syntactic parsing and shallow topic classification), scored at multiple levels (e.g. acoustic, semantic and context-based), and bid for by multiple agents, with the overall highest-confidence bid chosen.

The CDM provides a multi-device infrastructure, with customization to new applications and addition of plug-and-play devices eased by a declarative dialogue-move scripting language (Mirkovic and Cavedon, 2005). However, deciding which device an utterance is directed at is not always straightforward. One of our current application areas is a conversational interface to in-car devices, including entertainment, restaurant recommendation, navigation and telematic systems (Weng et al., 2004); in such an environment, a request such as *"Play X"* might be directed at an MP3 player or a DVD player. Eye-gaze (useful in multi-human dialogue) is not available, and we cannot rely on explicit device naming. One option is to use the resolution of NP arguments as disambiguating information (in our *"Play X"* example, whether X is a song or a movie). However, the NP-resolution process itself is often device-specific (see below), preventing NPs from being properly resolved until device has been determined.

Our proposed solution, inspired by approaches to multi-agent task allocation such as Contract Net (Smith, 1980), is to allow all devices to perform shallow processing of the incoming utterance, each producing multiple possible candidate dialogue moves. Potential device-move combinations are then scored against a number of features, including speech-recognition and parse confidence, discourse context, current *device-under-discussion*, and NP argument analysis. The device associated with the highest-scoring dialogue move is given first option to process the utterance. A disambiguation question may be generated if no device is a clear winner, or a confirmation question if the winning bid is not scored high enough.

Device choice, move choice, and selection of best ASR/parser hypothesis are thereby made simultaneously, rather than being treated as independent processes. As well as allowing for principled device identification, this has the benefit of scoring hypotheses on the basis of multiple information sources, including context. The highest scoring result overall may not correspond to the highest-confidence result from the ASR or parser n-best list alone, but n-best lists are effectively re-ordered based on device and dialogue context, allowing parsing errors such as incorrect PP-attachment to be automatically corrected. Confirmation and clarification behaviour can also be governed not only by ASR or parse confidence, but by the overall score.

* This work was performed while all the authors were employed at CSLI, Stanford University, and was partially supported by the US government's NIST Advanced Technology Program.

Related Approaches Rayner et al. (1994) combine speech recognition confidence scores with various intra-utterance linguistic features to re-order n-best hypotheses; Chotimongkol and Rudnicky (2001) also include move bigram statistics. Walker et al. (2000) use similar feature combination to identify misrecognised utterances. More recently, Gabsdil and Lemon (2004) also include pragmatic information such as NP resolution, and simultaneously choose from an n-best list while identifying misrecognition. They also divide misrecognised utterances into two overall confidence ranges, one for outright rejection and one for confirmation/clarification. Similarly Gabsdil and Bos (2003) combine acoustic confidences with semantic information, and Schlangen (2004) with bridging reference resolution, in order to allow clarification on an integrated basis. All of these approaches assume a single-device setting and hence no ambiguity of move type once the correct word string or parse has been identified. Here we extend these approaches to allow a principled choice of move/device pairing.

2 Background

2.1 Dialogue Manager Architecture

Our focus is on *activity-oriented dialogue*, discussing tasks or activities that are jointly performed by a human and one or more intelligent devices or agents. By "joint activity", we mean that the human participates in specifying the activity, clarifying requests, interpreting observations, and otherwise supporting the agent in the performance of the activity. Systems engaging in such dialogue characteristically require deep knowledge about the task domain and the devices/agents they provide access to, in order to know what information is critical to the tasks, and know what information about task performance is appropriate to provide to the user. CSLI has been developing activity-oriented dialogue systems for a number of years, for applications such as multimodal control of robotic devices (Lemon et al., 2002), speech-enabled tutoring systems (Fry et al., 2001), and conversational interaction with in-car devices (Weng et al., 2004).

The dialogue system architecture (Figure 1) centers around the CSLI Dialogue Manager, which can be used with various different external components: speech-recognizer, NL parser, NL generation, speech-synthesizer, as well as connections to external application-specific components such as ontologies or knowledge bases, and the dialogue-enabled devices themselves. Clean interfaces and representation-neutral processes enable the CDM to be used relatively seamlessly with different NL components, while interaction with external devices is mediated by *Activity Models*, declarative specifications of device capabilities and their relationships to linguistic processes.

The CDM uses the *information-state update* (ISU) approach to dialogue management (Larsson and Traum, 2000). The ISU approach extends the more traditional finite-state-based approaches used for simple dialogues (in which dialogue context is represented as one of a finite number of states, and each dialogue move results in a state transition), maintaining a richer representation of information-state. This includes the dialogue context as well as e.g. device and activity status, together with a set of update rules defining the effect of dialogue moves on the state (e.g. adding new information and referents for anaphora resolution, and triggering new tasks, activities and system responses). This approach allows more complex dialogue types with advanced strategies for context-dependent utterance interpretation (including fragments and revisions), NP resolution, issue tracking and improved speech-recognizer performance (Lemon and Gruenstein, 2004).

2.2 The CSLI Dialogue Manager

Generic ISU toolkits (e.g. TrindiKit (Traum et al., 1999), DIPPER (Bos et al., 2003)) provide general data structures for representing state and a language for specifying update rules, but the specific state and rules used are left to the individual application. The CDM is a specific implementation of an ISU dialogue-management system, providing data structures and processes for update specifically designed as suitable to activity-oriented dialogue, but adaptible to different applications and domains.

The two central components of the CDM information state are the *Dialogue Move Tree* (DMT) and the *Activity Tree*. The DMT represents the dialogue context and history, with each dialogue move represented as a node in the tree, and incoming moves interpreted in context by attachment to an appropriate open parent node (for example, `WhAnswer` moves attach to their corresponding `WhQuestion` nodes). This tree structure specifically supports *multi-threaded, multi-topic* conversations (Lemon et

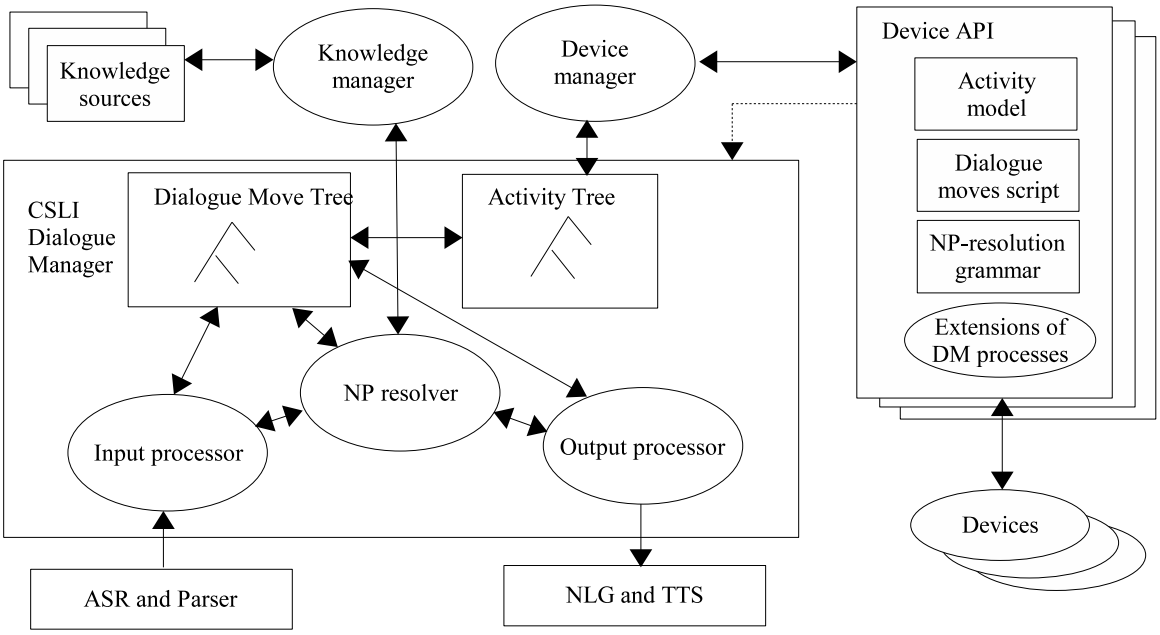

Figure 1: Dialogue System Architecture

al., 2002), with branches representing topics or threads: a dialogue move that cannot attach itself to the most recent active node may instead attach to another open branch (corresponding to a resumed conversation) or open a new branch (a new conversation thread) by attaching itself to the root node. The DMT also serves as context for interpreting fragments, multi-utterance constructs, and revisions, and provides discourse structure for tasks such as NP-resolution. In tandem, the Activity Tree manages the underlying activities, fully instantiating new activities via their Activity Models (e.g. resolving NP referents or spawning subdialogues to fill missing arguments), editing existing ones as a result of revisions or corrections, and monitoring their execution (possibly generating system moves notifying completion or failure).

Other data structures that are part of the information state include: the *salience list* (NPs and their referents for anaphora resolution); *multimodal input buffers* (semantic interpretations of GUI events); and the *system agenda* (potential system outputs scheduled by the dialogue manager). See (Lemon et al., 2002) for details.

2.3 Dialogue Move Scripting

In early versions of the CDM, dialogue moves were coded completely programmatically (in Java). While libraries of general-purpose dialogue moves (e.g. `Command`, `WhQuestion`, etc.) were re-used wherever possible, customization to new domains generally required significant programming effort in defining both new dialogue moves and their effects, and processes such as reference resolution. More recently, Mirkovic and Cavedon (2005) describe a dialogue-move scripting language designed to expedite customization to new domains. Each script serves a number of purposes:

1. hierarchical definition of dialogue moves, allowing inheritance and re-use of existing moves, while allowing customization to a specific domain;
2. mapping of utterance representations to appropriate dialogue moves, including argument values for devices' activity models;
3. definition of attachment rules for information-state update;
4. dialogue move-specific specification of output to be generated, for disambiguation or requests for required information.

Listing 1 shows the skeleton of a sample dialogue-move script for a `play Command` move for an MP3 player. The specific syntax of the *Input* and *Output* fields can be ignored for now: they simply match the interfaces of the parser and generator respectively. Variables in the dialogue move script correspond to variables in the Activity Model (AM) for the corresponding device. The AM for the MP3 device contains

```
User Command:play {  // inherits from generic Command dialogue move
   Description "play something"
   Input {           // templates for matching parser output
      // full parse match: ``play/start X''
      1.0   SYN{ s( features(mood(imperative)), predicate(#play/vb|#start/vb),
                 ?arglist(obj:_playable-object,?sbj:*)) }
      // full parse match: ``I want to play/hear X''
      1.0   SYN{ s( features(mood(indirect)), predicate(#play/vb|#hear/vb),
                 ?arglist(obj:_playable-object,?sbj:*)) }
      // topic classifier match
      0.1   TOPIC{ play_item }
      // topic classifier match with argument
      0.25  AND{ TOPIC{ play_item },   SYN{ arglist(obj:_playable-object,*) } }
   ... }
   Producing {       // templates for system output: questions
      System WHQuestion:disambiguate
      System WHQuestion:fill:play:_playable-object {
         Output {avs (e1 / play :question (q1 / what) :agent I)
         }
   ... }             // templates for system output: reports
         CloseOn System Report:play:playing {
            Output {avs (e1 / play :patient (p1 / [song]) :aspect continuous)
         }
   ... }
... }
```

Listing 1: Sample dialogue move script for a `play Command` for an MP3 device

a `play` operation with a (required) *playable-object* argument. When an incoming utterance matches an *Input* template from Listing 1, the *playable-object* variable is filled by unification, and resolved to an object from the device's domain which then fills the corresponding slot in the activity. For details, see (Mirkovic and Cavedon, 2005).

2.4 Multi-Device Dialogue

The CDM has also been extended to multi-device dialogue, with the scripting approach allowing easy dynamic plug-and-play specification of new "dialogue-enabled" devices. Note that this does not constitute multi-party dialogue: interaction is still mediated by a single dialogue manager, between a user and a *Device Manager* with which devices register themselves. However, the plug-and-play requirement (necessitated by the in-car application (Weng et al., 2004)) has resulted in important extensions to the dialogue management infrastructure.

Mirkovic and Cavedon (2005) describe a framework for encapsulating devices with information required to "dialogue-enable" them. Each device has associated with it the following components:

1. a set of dialogue-move scripts;
2. an Activity Model describing any device functionality accessible by dialogue;
3. a device-specific ontology and knowledge base (KB);
4. rules for device-specific NP-resolution.

Any significantly different forms of interaction requiring device-specific dialogue management processes must still be specified as new Java classes (referred to as *DM process extensions* in Figure 1), but in general the above four components contain the device-specific information required for dialogue-enabling new devices.

Note that NP resolution rules are included in the device definition; while pronoun resolution tends to be domain-independent, resolving definite descriptions and demonstratives is often device-dependent, and resolving named referents often requires constructing appropriate queries to a device-specific knowledge-base.

Devices can now be added dynamically to the DMT, registering themselves with the Device Manager and becoming associated with their own nodes to which new conversation threads can attach; "current device" becomes part of the information-state and interpreting incoming utterances is performed in this context.

In this context, device selection—determining which device an utterance is associated with—becomes a further complication: an utterance may (on the surface) be potentially applicable to multiple devices: e.g. "Play *X*" could be applicable to either an MP3 player or a DVD

player. Our original proposal was to create a dialogue move consistent with each such device and then score its applicability based on other factors, e.g. ability to resolve the object reference (the MP3 player would resolve a song-name, the DVD player a movie name). The rest of the paper generalises this approach to a wider range of possible disambiguities, involving a greater number of scoring features, and results in more interesting behaviours than simple device-disambiguation.

3 Multiple Interpretation Methods

The first new extension to the CDM described here is the use of multiple information sources in parallel to classify dialogue move type and produce an activity-specific representation. In most systems (and previous incarnations of the CDM) a single interpretation mechanism is chosen which is best suited to the application at hand, be it e.g. an open-domain statistical parser, a domain-specific constraint-based grammar, or keyword-spotting techniques. We extend this approach here to allow arbitrary multiple interpretation mechanisms, each producing its own (independent) interpretation hypothesis and associated confidence. In the current application, we use both a statistical parser producing relatively deep dependency structures, and a shallow maximum-entropy-based topic classifier.

Dialogue move scripts, such as the one sketched in Listing 1, are used to construct instantiations of candidate dialogue moves for a device, based on incoming user utterances (and planned system outputs, although we focus on the former here). This is governed by the *Input* field for each move type, which specifies a set of patterns: when an utterance representation matches an *Input* pattern, a candidate node of the appropriate type can be created. As Listing 1 shows, patterns can now be defined in terms of interpretation method as well as the interpreted form itself: `SYN` patterns match the output of the statistical parser, `TOPIC` patterns match the output of the topic classifier, while `AND` patterns match combinations of the two. Further general pattern types are available (e.g. `LF` for semantic logical forms, `STRING` for surface string keyword-matching) but are not used in the current application.

Each pattern is associated with a weight, used in the overall move scoring function described in Section 4 below. This allows moves created from matches against deep structure to be scored highly (e.g. `SYN` patterns in which predicate and arguments are specified and matched against), shallow matches to be scored low (e.g. simple `TOPIC` matches), and combined matches to have intermediate scores (e.g. a combination of an appropriate `TOPIC` classification with a `SYN` parser output containing a suitable NP argument pattern). Depending on other elements of the scoring function (e.g. the ASR confidence associated with the hypothesised string being tested) and on competing move hypotheses, low scores may lead to clarification being required (and therefore clarification will be more likely when only low-scoring (shallow) patterns are matched). Behaviour can therefore be made more robust: when deep parsing fails, a shallow hypothesis can be used instead (clarifying/confirming this specific hypothesis as necessary depending on its confidence) rather than resorting to a rejection or general clarification. Scores are currently set manually and determined by testing on sample dialogues; future work will examine learning them from data.

4 Dialogue Move Selection

In the general case, multiple possible candidate dialogue moves will be produced for a given user utterance, for a number of reasons:

1. multiple hypotheses from ASR/parser output;
2. multiple interpretation methods (deep parsing vs. shallow classification);
3. multiple possible move types for a candidate interpretation;
4. multiple antecedent nodes (active dialogue threads), including multiple devices, for a particular move type.

These are not independent: it is important to consider all factors simultaneously, to allow an integrated scoring function for each candidate and thus consider the best overall. The skeleton algorithm for instantiating and selecting a dialogue move is therefore as follows:[1]

[1]Note that we will not create $O \times N \times M$ candidates: only a subset of script entries (if any) will match for each node and n-best entry.

```
foreach open node O
   foreach n-best list entry N
      foreach matching script entry M
         create candidate move
score all candidates
if (score(top) >> score(second))
then
   select top candidate
else
   generate question to disambiguate
if (score(selected-node) < threshold)
    generate question to confirm
```

The interesting aspect of the above process is the scoring function. Dialogue-move candidates are scored using a number of weighted features, ranging from speech-recognizer confidence, through to pragmatic features such as the "device in focus" and recency of the DMT node the candidate would attach to. The full list of features currently considered is shown in Table 1. Note the inclusion of features at many levels, from acoustic recognition confidences through syntactic parse confidence to semantic and pragmatic features.

4.1 Reordering n-best candidates

This integrated scoring mechanism therefore allows n-best list input to be re-ordered: dialogue-move candidates are potentially instantiated for each n-best list entry and the highest-scoring candidate chosen. While the n-best list rank and confidence are factors in the overall score, other features may outweigh them, resulting in an initially lower-ranked n-best entry becoming the highest-scoring dialogue move.

Evaluation so far has been limited to initial testing on a manually constructed set of test inputs, using only a subset of the features: those shown italicised in Table 1 are not currently available due to either implementational issues (for full domain referent resolution and KB queries) or lack of domain data (for move bigram frequencies). Our test set includes 400 sentences, of which 300 have been used in training the statistical parser and 100 are unseen variations; it currently covers only utterances related to a single device (a restaurant recommendation system) and does not include speech recognition hypotheses (we are therefore testing parse n-best reordering only). We are currently working towards evaluation on a full set of features, with user-generated multi-device speech input.

However, even with the restricted set of features, preliminary testing on this set shows encouraging results: the percentage of sentences for which the correct parse is chosen increases from 90% to 94%, a 41% reduction in error with several common parse errors being corrected. One example is incorrect PP-attachment (a notoriously difficult challenge for statistical parsers). The example below (from a restaurant recommendation scenario), shows the top two n-best list entries for a sentence as produced by our statistical parser:

```
1. how about [a restaurant
              [in Grant]] [on Mayfield]
2. how about [a restaurant
              [in Grant] [on Mayfield]]
```

Here, the second is lower-ranked but correct, taking both PPs as modifying *restaurant*, while the first treats only one as modifying *restaurant*, one as a sentential modifier. As the second allows two database-query constraints to be filled (city and street name), and the first just one, this boosts its overall score enough to overcome its lower parse confidence, and it is selected and used in DMT attachment. Similar improvements are gained with nominal modifiers:

```
1. how about [a
           [[cheap] chinese] restaurant]
2. how about [a
           [cheap] [chinese] restaurant]
3. how about [a
           [cheap chinese] restaurant]
```

Here the second is correct, treating *cheap* and *chinese* as both independently modifying *restaurant*; the first takes *cheap* as modifying *chinese*, and the third takes *cheap chinese* as a single multi-word unit. Again, as the second fills two database-query constraints (price level and cuisine type), its overall score becomes highest. Evaluation of the improvement achieved is currently in progress.

4.2 Move type comparison

The scoring function for feature combination is currently manually defined. When comparing between candidate moves of the same type, this is relatively straightforward, although hardly trivial and inherently done to a high extent by subjective expertise. However, it becomes much less straightforward when comparing candidates of different types, as some move types and some DMT attachment contexts will allow only a subset of the features to have meaningful values. However, comparison between move types is essential, as two ASR hypotheses with

Recognition features:	recognition and parse probabilities; recognition and parse n-best ranks;
Semantic features:	topic classification for the parse (with score); for dialogue moves spawning activities: - number of slots filled by input pattern; - *number of resolved/unresolved slots after NP resolution;* - *number of ambiguously resolved slots after NP resolution;* for queries about database objects: - set of constraints sent to the knowledge base; - *cardinality of the set of knowledge base query results;*
Contextual features:	current most active node; current activity; position and recency of the parent node in the active node list; *bi-gram frequencies of the dialogue moves:* - *DMT attachments - pairs of child and parent node types;* - *pairs of chronologically consecutive user nodes.*

Table 1: Move Scoring Features

similar recognition scores may have very different possible move types:

1. Command: "Play a rock song by Cher"
2. Query: "What rock songs are there?"

We are therefore currently investigating the use of machine learning techniques to improve on our current manual definitions. With annotated data the optimal weights of a scoring function that combines all the features can be automatically learned (see (Gabsdil and Lemon, 2004)).

4.3 Dialogue-move disambiguation

In order for a winning bid to be unambiguously accepted, its score must exceed the next highest score by more than a predefined threshold. If not, we take the choice of winning bid to be within our margin of error, and the dialogue manager asks a disambiguating clarification question. For example, if the pair of sentences in the previous section result in hypothesis dialogue moves with scores within the margin of error, then the dialogue manager generates a question of the form:

> "Did you want to play a rock song by Cher or did you ask about rock songs?"

Alternatively, in some cases there may be a clear highest-scoring bid (i.e. one of high *relative* confidence) which is itself of low *absolute* confidence. In such cases, rather than act on the move unconditionally we ask the user for clarification. If the score is below a certain confidence threshold T_1 we treat the highest bid as a reasonable hypothesis, but ask for confirmation of the intended move; following the previous example, this would result in a question such as:

> "I'm not sure I understood that. Did you want to play a rock song by Cher?"

If the score is below a second critical minimum threshold T_2 we take this as a failure in interpretation, and prompt for general clarification. As even the best hypothesised move is likely to be incorrect in this case (being of such low confidence), asking for specific confirmation is likely to be counter-productive or annoying (see e.g. (San-Segundo et al., 2001)).

Threshold values are currently specified as part of dialogue-move definitions; a future direction is to automatically learn optimal values for the thresholds.

5 Discussion and Conclusions

We have described a number of strategies implemented in the CSLI Dialogue Manager to more robustly handle ambiguous or misunderstood utterances, and low-confidence interpretations. Features from multiple sources of evidence are combined to rate the possible dialogue move candidates as interpretations of a user utterance. Features include confidence scores from ASR and parser, as well as semantic and pragmatic criteria, and measures related to the dialogue context itself. As well as selecting dialogue move, in our multi-device setting the approach has the benefit of selecting the device being addressed. Although we have not yet performed a full evaluation of the ef-

ficacy of this approach, we have observed several examples of the n-best list of inputs being (correctly) re-ordered—i.e. after misclassification by the statistical parser, the candidate dialogue-move corresponding to the correct (though lower-confidence) parse can still be selected. We are currently gathering data in order to provide a concrete evaluation.

Confidence thresholds (upper and lower bounds) set by the dialogue designer specify the levels at which a candidate move is rejected, requires explicit confirmation by the user, or simply accepted. Future work includes automatically learning optimal values for these thresholds and optimal weights on the features for scoring candidate dialogue-moves, applying the techniques of e.g. Gabsdil and Lemon (2004) to our multi-device setting.

References

J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and formalization of an information-state update dialogue system architecture. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*.

A. Chotimongkol and A. Rudnicky. 2001. N-best speech hypotheses reordering using linear regression. In *Proceedings of the 7th European Conference on Speech Communication and Technology (EUROSPEECH)*.

J. Fry, M. Ginzton, S. Peters, B. Clark, and H. Pon-Barry. 2001. Automated tutoring dialogues for training in shipboard damage control. In *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*.

M. Gabsdil and J. Bos. 2003. Combining acoustic confidence scores with deep semantic analysis for clarification dialogues. In *Proc. 5th International Workshop on Computational Semantics (IWCS-5)*.

M. Gabsdil and O. Lemon. 2004. Combining acoustic and pragmatic features to predict recognition performance in spoken dialogue systems. In *Proc. 42nd Annual Meeting of the ACL*.

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

O. Lemon and A. Gruenstein. 2004. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3).

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

D. Mirkovic and L. Cavedon. 2005. Practical plug-and-play dialogue management. In *Proceedings of the Annual Meeting of the Pacific Association of Computational Linguistics (PACLING)*.

M. Rayner, D. Carter, V. Digalakis, and P. Price. 1994. Combining knowledge sources to reorder n-best speech hypothesis lists. In *Proceedings of the ARPA Human Language Technology Workshop*.

R. San-Segundo, J. M. Montero, J. Ferreiros, R. Córdoba, and J. M. Pardo. 2001. Designing confirmation mechanisms and error recover techniques in a railway information system for spanish. In *Proc. 2nd SIGdial Workshop on Discourse and Dialogue*.

D. Schlangen. 2004. Causes and strategies for requesting clarification in dialogue. In *Proc. 5th SIGdial Workshop on Discourse and Dialogue*.

R. G. Smith. 1980. The contract net protocol: High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

D. Traum, J. Bos, R. Cooper, S. Larsson, I. Lewin, C. Matheson, and M. Poesio. 1999. A model of dialogue moves and information state revision. In *Task Oriented Instructional Dialogue (TRINDI): Deliverable 2.1*. University of Gothenburg.

M. Walker, J. Wright, and I. Langkilde. 2000. Using natural language processing and discourse features to identify understanding errors in a spoken dialogue system. In *Proceedings of the 17th International Conference on Machine Learning*.

F. Weng, L. Cavedon, B. Raghunathan, D. Mirkovic, H. Cheng, H. Schmidt, H. Bratt, R. Mishra, S. Peters, L. Zhao, S. Upson, E. Shriberg, and C. Bergmann. 2004. A conversational dialogue system for cognitively overloaded users. In *Proc. 8th International Conference on Spoken Language Processing (INTERSPEECH)*.