A Conversational Dialogue System for Cognitively Overloaded Users

Fuliang Weng¹ Lawrence Cavedon² Badri Raghunathan¹ Danilo Mirkovic²
Hua Cheng³ Hauke Schmidt¹ Harry Bratt³ Rohit Mishra⁴ Stanley Peters²
Lin Zhao¹ Sandra Upson¹ Liz Shriberg³ Carsten Bergmann⁴

¹Research and Technology Center, Robert Bosch Corp., Palo Alto, California, USA
²Center for the Study of Language and Information, Stanford University, California, USA
³Speech Technology and Research Lab, SRI International, Menlo Park, California, USA
⁴Electronics Research Lab, Volkswagen of America, Palo Alto, California, USA

{fuliang.weng,badri.raghunathan,hauke.schmidt}@rtc.bosch.com
{l.cavedon,danilom,huac,peters}@csli.stanford.edu
{harry,ees}@speech.sri.com
{rohit.mishra,carsten.bergmann}@vw.com

Abstract
Spoken dialogue interfaces are gaining increased acceptance in a wide range of applications. Most current examples of such systems, however, rely on using restricted language and scripted dialogue interactions. We argue that speech interfaces in highly stressed or cognitively overloaded domains, i.e. those involving a user concentrating on other tasks, call for more flexible dialogue with robust, wide-coverage language understanding. We describe an initial effort at addressing flexible and rich dialogue in a system with a number of features, such as full spoken language understanding, a multi-threaded dialogue manager, dynamic update of information, and recognition of partial proper names.

1. Introduction
Most current applications of spoken language dialogue systems involve narrowly focused language understanding and simple models of dialogue interaction. Real human dialogue, however, is highly context- and situation-dependent, full of ill-formed utterances and sentence fragments, and highly interactive and collaborative: speakers interrupt each other, finish each other’s sentences, and jointly build contributions to the shared context.

Understanding language and modeling natural dialogue of this form is important in building friendly spoken-language interfaces, but it is particularly critical in settings where the user is focused on external tasks, such as simultaneously operating multiple devices, or driving a car. In such scenarios, users cannot plan their utterances ahead of time or “hold that thought” until an appropriate time: they need to be able to interrupt the dialogue system, issue instructions that build on the context and situation, and the system must be able to interpret these contributions in context. Conversely, the dialogue system should only interrupt the user when appropriate (such as in critical situations), and any questions from the system should be as focused as possible.

This paper describes a joint project targeted at building such a spoken dialogue system for interacting with a variety of devices. In addition to a whole system description, we focus on two core components in the current system: a statistical language understanding (NLU) module that provides robust full sentence (and fragment) analysis; and a dialogue manager that supports situation- and context-dependent interpretation and manages user-system interaction. We illustrate capabilities with a simple dialogue involving control of an MP3 music-player (a domain with challenges in dealing with proper names).

Salient features of the implemented system include:
- Robustness in the face of imperfect input from the Speech Recognition (SR) module and human users, including incomplete references to partial proper names;
- Full language analysis: the system has to understand subtleties in non-content words, such as definite vs. indefinite determiners, and qualifiers;
- Dynamic updates: devices and information content may be added or extended dynamically;
- Situation- and context-dependent interpretation of user utterances, including non-sentential fragments.

This paper is organized as follows. Section 2 describes the overall system architecture. The next three sections describe the three core modules: the NLU module, the Dialog Management module, and the Knowledge Management module, together with their working functionalities. Section 6 gives a brief description of our data collection setup and forthcoming evaluation. We conclude with a description of the current status as well as ongoing and future work.

2. The conversational dialogue system
This section describes the general system architecture and communication framework. The main functionalities are illustrated via an actual sample dialogue for controlling an MP3 music player.

2.1. System architecture
The current system provides end-to-end spoken language processing for interaction with multiple devices, using a combination of best off-the-shelf components, components previously used in other language applications, and
components and techniques specifically developed as part of this project. Rather than the hub architecture employed by systems as developed in Communicator projects [10], we use an event-based, message-oriented middleware: such architectures are the current popular paradigm for distributed systems, especially those allowing dynamic registration of new components.

Among the component modules in Figure 1, we use the Nuance speech recognition engine with class-based n-grams and dynamic grammars, and Nuance Vocalizer as the TTS engine. The Prosody module (to be developed) will provide additional input to the NLU and DM modules to improve their performance. The core components are the NLU module, the DM module, and the Knowledge Manager, each of which is described in the following sections.

2.2. Example dialogue

The following short dialogue illustrates a range of phenomena and functionalities, taken from the MP3 domain, that the current system can handle, including NP resolution based on context and external situation, dynamic content updates, disambiguation, multi-threaded conversation, and complex context and external situation.

1. **User:** What music do you have?
   **System:** I have pop, rock and soul music.

2. **User:** What about some rock.
   **System:** I have pop, rock and soul music.

3. **User:** What songs are on Brothers in Arms?
   **System:** Brothers in Arms has the songs one world, why worry, so far away, <...>

4. **User:** What albums do you have by them?
   **System:** Brothers In Arms.

5. **User:** How is this?
   **System:** This is Dire Straits.

6. **User:** What's on Drivetime? <spawns subdialogue>
   **System:** combination, believe and love is the groove

7. **User:** Why worry.
   **System:** <starts playing why worry instead>

8. **User:** Add this to a playlist.
   **System:** Which playlist do you mean? I have Drivetime and WorkSongs <disambiguation>

9. **User:** What's on Drivetime? <spawns subdialogue>
   **System:** combination, believe and love is the groove

10. **User:** Add it to Drivetime.
    **System:** Ok, added money for nothing to playlist Drivetime.

11. **User:** You have 28 new songs.
    **System:** <KB updated dynamically; new songs can be accessed>

3. The NLU module

To provide maximum flexibility of spoken language while still achieving robustness, we designed and implemented a statistical natural language understanding (NLU) module. The NLU module consists of three sub-modules: a statistical tagger, a statistical dependency parser, and a feature mapper. The tagger uses Hidden Markov Models (HMM) to describe probabilities in predicting part of speech (POS) tags based on history [1], while the Viterbi algorithm is used to find the best tag sequence. The dependency parser uses the algorithm described in [13] with some simplifications. The new model consists of two components:

The conditional probability: this estimates the probability of the grammatical relation of two heads given their two corresponding subtrees. It is computed using the SGC algorithm for conditional maximum entropy modeling [15]:

\[
p(y | x) = \frac{1}{Z(x)} \exp(\sum_j \lambda_j f_j(x, y))
\]

where \( y \) is a grammatical relation, \( x \) represents the left and right subtrees as defined by a ten-dimensional array that contains head words, tags, and grammatical relations, and \( f_j(x, y) \). \( \lambda_j \), and \( Z(x) \) are the features, weights, and normalization factors, respectively.

The mutual information: this characterizes the redundancy between the representatives of the left and right subtrees, and it is computed through the factorization of the head tags and conditional mutual information of the words given tags.

The feature mapper takes the dependency relations from the statistical dependency parser and produces an XML representation similar to f-structure. In the representation, head-words, predicates, various modifiers, and sentential features, such as mood, aspect, and tense, are explicitly listed. The mapping is based on a set of patterns of the heads and the modifiers. The patterns may include words, tags, or grammatical relations.

Unlike in many current dialogue systems, particularly template-based ones [2,3], the NLU module provides robust
full analysis of an input sentence. Not only are conventionally perceived content words extracted, but so are the subtle meanings of various other words. For example, the articles “a” and “the” are differentiated by the NLU module and interpreted appropriately by the DM.

The NLU module supports dynamic updates of the knowledge base. The dependency parser is trained on template data. For example, in the MP3 domain, the training data does not contain any specific song names, but rather a generic class name called songname. This is true for other classes as well. During parsing, a new song database is supplied or updated as described above. When an input sentence contains a full song name, the name is first identified and its class-name songname is added to the existing word sequence as an alternative path. A more difficult case is when song names are incomplete (a common case with distracted users). A recognizer for partial proper names has been developed that labels the corresponding word sub-sequence. The algorithm reaches more than 90% accuracy when the song name database is given [14]. The modified input lattice with proper names is then given to the parser for analysis. This is essentially a statistical version of the virtual terminal technique used in grammar partitioning and parser composition [12].

4. Dialogue management module

Interaction is mediated and managed by the CSLI Dialogue Manager (CDM). The CDM uses the dialogue-move approach [6] to maintain dialogue context, which is then used to interpret incoming utterances (including fragments and revisions), resolve NPs, construct salient responses, track issues, etc. Dialogue state can also be used to bias SR expectation and improve SR performance, as has been performed in previous applications of the CDM. Detailed descriptions of the CDM can be found in [7].

4.1. Multi-domain design

The CDM is designed to be multi-domain and has been applied to a range of applications, including control of intelligent devices [7] and tutoring [4]. Different application domains typically involve specifying a different grammar for surface-level generation, and a device-specific Activity Model. The Activity Model is a declarative specification of the capabilities of the agent or device with which the CDM interacts, and includes linguistic information, such as mappings from verbs and their arguments to device-actions. The arguments that are marked as required may generate sub-dialogues when a user-command is given with missing arguments.

The current version of the dialogue manager includes a Semantic Analyzer component. This component constructs semantic logical forms from a quasi f-structure obtained from the NLU module and classifies them by dialogue-move type (e.g., command, wh-question) using dialogue-context information. Some of the internal CDM components allow a mix of domain-dependent and domain-independent processes, where default algorithms are implemented independently from a particular domain, but some of their sub-processes can be easily configured to make use of domain-dependent

1 We plan to incorporate prosodic information into this task.

information. Similarly, for new applications, the current implementation also allows the expansion of logical forms or dialogue-move operators without disrupting the core CDM implementation.

4.2. Multi-threaded context management

The CDM has been specifically designed to manage multi-threaded, multi-topic conversations [7]. This is particularly important in the car situation, where the driver interacts with multiple devices, potentially interleaved. For example, an interaction to choose and book a restaurant may be interrupted by a cell phone request. The two central CDM components for supporting multi-threaded conversation are the Dialogue Move Tree (DMT) and the Activity Tree (AT).

The DMT represents the historical context of a dialogue. An incoming utterance, classified as a 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. A new conversation topic spawns a new branch; a dialogue move that cannot attach itself to the most recent active node may attach to an active node in another branch, which corresponds to a resumed conversation.

The AT manages activities relevant to a dialogue. When the user issues a command, this generally results in a new activity being created and added to the AT. Before the activity can actually be sent to the device for execution, the system attempts to fully resolve it, e.g., resolving all NPs or spawning a sub-dialogue to elicit further information. Revisions and corrections (e.g. “I meant/said …”) typically involve editing an existing activity representation. Activity-execution is monitored on the AT and changes may result in generated output, e.g. on failure or successful completion.

The DMT and AT act as a framework for other important dialogue context functionalities, such as generating task reports and grounding the user after a break in conversation (“What were we talking about?”) [5]. The DMT also provides implicit discourse structure for tasks such as reference resolution.

5. Knowledge management module

The Knowledge Manager (KM) controls access to knowledge base sources (such as domain knowledge and device information) and their updates. Domain knowledge is structured according to domain-dependent ontologies. The current KM makes use of Protégé, a domain-independent ontology tool. The CDM queries the KM for instances matching semantic descriptions constructed from a command or query. For example, in the MP3 domain, a command to “play some rock music by Cher” results in a query for objects of class song with genre=rock and artist=Cher, where genre and rock are (inherited) properties of the class song. When many results satisfy constraints from the user, the DM uses the ontology hierarchy to categorize them and output them in a succinct way to reduce the user’s cognitive load (as in the sample dialogue). As mentioned above, the KB can be dynamically updated with new instances at any point.

2 Multi-device interaction is via a central controller, which seems to be preferred by users [8].

3 http://protege.stanford.edu
In addition, the KM also serves as the repository of device information, such as the Activity Model. As a new device is made available, it registers its information with the KM, which then makes it available to the CDM.

6. Data collection setup

To improve and evaluate system performance, we have started to collect dialogue data for our task. We are using the “Wizard of Oz” (WOZ) approach, where an experiment subject talks to a (unseen) person in another room pretending to be a smart machine. The WOZ approach allows eliciting high quality dialogue while setting correct expectations for the subject in terms of language complexity, and avoids the need for automatically understanding unconstrained language.

To simulate a cognitively overloaded scenario, we use a driving simulator as an occupying task. As part of the simulator, we use the video game Midtown Madness 1, which is situated in downtown Chicago. A steering wheel with force feedback is mounted in front of the game display. The wizard (in a separate room) acts as an “ideal” dialogue system and the subject interacts with the wizard without knowing she is a person. The wizard sees the game screen through a video splitter so that she knows the subject’s current situation in the driving scene. When the subject needs navigation or entertainment information, she presses a button on the wheel and talks into a head-mounted microphone; the wizard only hears the subject speech when this button is pressed.

The speech from both subject and wizard is recorded to a hard disk by our recording software. We designed one scenario for operating the MP3 player, where the subjects are asked to create two song lists while they are driving around the city. To create song lists, the subjects may query about songs in their collection, listen to songs, and add to or delete songs from their lists; this scenario provides multi-threaded dialogues. Other scenarios increase cognitive load by adding time-constraints on driving tasks. We have observed 20 subjects and recorded their speech. We are in the process of transcribing the speech and annotating for dialogue information.

7. Current status and future work

The current system is built using the generic architecture described in Section 3 and supports the example dialogue provided in Section 2 for interacting with an MP3 player. Evaluation across a number of metrics, relating to speech recognition rates, language understanding, and successful task completion, are planned for the current system. We expect to be able to report on these results by the time of the conference. The current stage of development is proceeding in a number of directions, including:

- Techniques for robust interpretation, especially of non-sentential fragments;
- Defining new devices, including telematic services (e.g. restaurant selection). The CDM will be extended to handle dynamic registration of new devices, extending the “plug-and-play” techniques of [9];
- Using prosodic information for improved SR, NLU, and dialogue-move classification [11];
- Implementing user-modelling and situation-awareness for adaptive NL generation;
- Improving SR using context-dependent biasing and noise cancellation techniques.

8. Acknowledgements

This project is partially supported by a NIST ATP project, Robert Bosch Corp., and VW of America.

9. References