University of Colorado Dialog Systems for Travel and Navigation

B. Pellom, W. Ward, J. Hansen, R. Cole, K. Hacioglu, J. Zhang, X. Yu, S. Pradhan

Center for Spoken Language Research, University of Colorado
Boulder, Colorado 80303, USA
{pellom, whw, jhlh, cole, hacioglu, zjp, xiu, spradhan}@cslr.colorado.edu

ABSTRACT

This paper presents recent improvements in the development of the University of Colorado "CU Communicator" and "CU-Move" spoken dialog systems. First, we describe the CU Communicator system that integrates speech recognition, synthesis and natural language understanding technologies using the DARPA Hub Architecture. Users are able to converse with an automated travel agent over the phone to retrieve up-to-date travel information such as flight schedules, pricing, along with hotel and rental car availability. The CU Communicator has been under development since April of 1999 and represents our test-bed system for developing robust human-computer interactions where reusability and dialogue system portability serve as two main goals of our work. Next, we describe our more recent work on the CU Move dialog system for in-vehicle route planning and guidance. This work is in joint collaboration with HRL and is sponsored as part of the DARPA Communicator program. Specifically, we will provide an overview of the task, describe the data collection environment for in-vehicle systems development, and describe our initial dialog system constructed for route planning.

1. CU COMMUNICATOR

1.1 Overview

The Travel Planning Task

The CU Communicator system [1,2] is a Hub compliant implementation of the DARPA Communicator task [3]. The system combines continuous speech recognition, natural language understanding and flexible dialogue control to enable natural conversational interaction by telephone callers to access information from the Internet pertaining to airline flights, hotels and rental cars. Specifically, users can describe a desired airline flight itinerary to the Communicator and use natural dialog to negotiate a flight plan. Users can also inquire about hotel availability and pricing as well as obtain rental car reservation information.

System Overview

The dialog system is composed of a Hub and several servers as shown in Fig. 1. The Hub is used as a centralized message router through which servers can communicate with one another [4]. Frames containing keys and values are emitted by each server, routed by the hub, and received by a secondary server based on rules defined in a "Hub script".

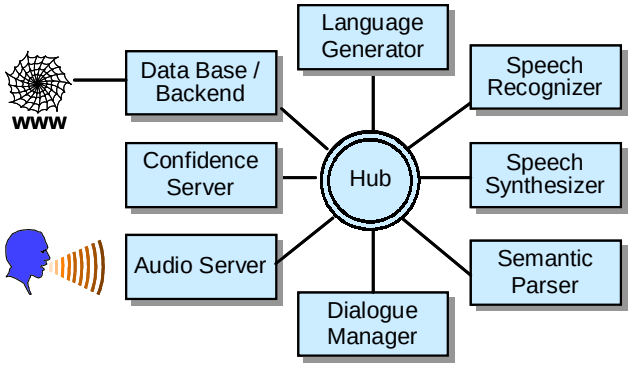

Figure 1. Block diagram of the functional components that comprise the CU Communicator system[1].

1.2 Audio Server

The audio server is responsible for answering the incoming call, playing prompts and recording user input. Currently, our system uses the MIT/MITRE audio server that was provided to DARPA Communicator program participants. The telephony hardware consists of an external serial modem device that connects to the microphone input and speaker output terminals on the host computer. The record process is pipelined to the speech recognition server and the play process is pipelined the text-to-speech server. This audio server does not support barge-in.

Recently we have developed a new audio server that supports barge-in using the Dialogic hardware platform. The new audio server implements a Fast Normalized Least-Mean-Square (LMS) algorithm for software-based echo cancellation. During operation, the echo from the system speech is actively cancelled from the recorded audio to allow the user to cut through while

[1] This work was supported by DARPA through SPAWAR under Grant No. N66001-002-8906. The "CU Move" system is supported in part through a joint collaboration with HRL Laboratories.

the system is speaking. The new audio server operates in the Linux environment and is currently being field-tested at CSLR. Because the server implements software-based echo cancellation, it can work on virtually any low-cost Dialogic hardware platform. This server will be made available to the research community as a resource in the near future.

1.3 Speech Recognizer

We are currently using the Carnegie Mellon University Sphinx-II system [5] in our speech recognition server. This is a semi-continuous Hidden Markov Model recognizer with a class trigram language model. The recognition server receives the input vectors from the audio server. The recognition server produces a word lattice from which a single best hypothesis is picked and sent to the hub for processing by the dialog manager.

Acoustic Modeling

During dialog interaction with the user, the audio server sends the acoustic samples to three Sphinx-II speech recognizers. While the language model is the same for each decoder, the acoustic models consist of (i) speaker independent analog telephone, (ii) female adapted analog telephone, and (iii) cellular telephone adapted acoustic model sets. Each decoder outputs a word string hypothesis along with a word-sequence probability for the best path. An intermediate server is used to examine each hypothesis and pass the most likely word string onto the natural language understanding module.

Language Modeling

The Communicator system is designed for end users to get up-to-date worldwide air travel, hotel and rental car information via the telephone. In the task there are word lists for countries, cities, states, airlines, etc. To train a robust language model, names are clustered into different classes. An utterance with class tagging is shown in Fig.2. In this example, *city*, *hour_number*, and *am_pm* are class names.

Original Utterance

I want to go from Boston to Portland around nine a_m

Class-Tagged Utterance

I want to go from [city:Boston] to [city:Portland] around [hour_number:nine] [am_pm:a_m]

Concept-Tagged Utterance

[I_want: I want to go] [depart_loc: from Boston] [arrive_loc: to Portland] [time:around nine a_m]

Figure 2. Examples of class-based and grammar-based language modeling

Each commonly used word takes one class. The probability of word W_i given class C_i is estimated from training corpora. After the corpora are correctly tagged, a back-off class-based trigram language model can be computed from the tagged corpora. We use the CMU-Cambridge Statistical Language Modeling Toolkit to compute our language models.

More recently, we have developed a dialog context dependent language model (LM) combining stochastic context free grammars (SCFGs) and n-grams [6,7]. Based on a spoken language production model in which a user picks a set of concepts with respective values and constructs word sequences using phrase generators associated with each concept in accordance with the dialog context, this LM computes the probability of a word, P(W), as

$$P(W) = P(W/C)\, P(C/S) \qquad (1)$$

where W is the sequence of words, C is the sequence of concepts and S is the dialog context. Here, the assumptions are (i) S is given, (ii) W is independent of S but C, and (iii) W and C associations are unambiguous. This formulation can be considered as a general extension of the standard class word based statistical language model as seen in Fig. 2.

The first term in (1) is modeled by SCFGs, one for each concept. The concepts are classes of phrases with the same meaning. Each SCFG is compiled into a stochastic recursive transition network (STRN). Our grammar is a semantic grammar since the nonterminals correspond to semantic concepts instead of syntactic constituents. The set of task specific concepts is augmented with a single word, multiple word and a small number of broad but unambigious part of speech (POS) classes to account for the phrases that are not covered by the grammar. These classes are considered as "filler" concepts within a unified framework. The second term in (1) is modeled as a pool of concept n-gram LMs. That is, we have a separate LM for each dialog context. At the moment, the dialog context is selected as the last question prompted by the system, as it is very simple and yet strongly predictive and constraining. SCFG and n-gram probabilities are learned by simple counting and smoothing. Our semantic grammars have a low degree of ambiguity and therefore do not require computationally intensive stochastic training and parsing techniques.

Experimental results with N-best list rescoring were found promising (5-6% relative improvement in WER). In addition, we have shown that a dynamic combining of our new LM and the standard class word n-gram (the LM currently in use in our system) should result in further improvements. At the present, we are interfacing the grammar LM to the speech recognizer using a word graph.

1.4 Confidence Server

Our prior work on confidence assessment has considered detection and rejection of word-level speech recognition errors and out-of-domain phrases using language model features [8]. More recently [9], we have considered detection and rejection of misrecognized units at the concept level. Because concepts are used to update the state of the dialog system, we believe that concept level confidence is vitally important to ensuring a graceful human-computer interaction. Our current work on concept error detection has considered language model features (e.g., LM back-off behavior, language model score) as well as acoustic features from the speech recognizer (e.g., normalized acoustic score, lattice density, phone perplexity). Confidence

features are combined to compute word-level, concept-level, and utterance-level confidence scores.

1.5 Language Understanding

We use a modified version of the Phoenix [10] parser to map the speech recognizer output onto a sequence of semantic frames. A Phoenix frame is a named set of slots, where the slots represent related pieces of information. Each slot has an associated context-free semantic grammar that specifies word string patterns that can fill the slot. The grammars are compiled into Recursive Transition Networks, which are matched against the recognizer output to fill slots. Each filled slot contains a semantic parse tree with the slot name as root.

Phoenix has been modified to also produce an extracted representation of the parse that maps directly onto the task concept structures. For example, the utterance

"I want to go from Boston to Denver Tuesday morning"

would produce the extracted parse:

Flight_Constraint: Depart_Location.City.Boston
Flight_Constraint: Arrive_Location.City.Denver
Flight Constraints:[Date_Time].[Date].[Day_Name].tuesday
[Time_Range].[Period_Of_Day].morning

1.6 Dialog Management

The Dialogue Manager controls the system's interaction with the user and the application server. It is responsible for deciding what action the system will take at each step in the interaction. The Dialogue Manager has several functions. It resolves ambiguities in the current interpretation; Estimates confidence in the extracted information; Clarifies the interpretation with the user if required; Integrates new input with the dialogue context; Builds database queries (SQL); Sends information to NL generation for presentation to user; and prompts the user for missing information.

We have developed a flexible, event driven dialogue manager in which the current context of the system is used to decide what to do next. The system does not use a dialogue network or a dialogue script, rather a general engine operates on the semantic representations and the current context to control the interaction flow. The Dialogue Manager receives the extracted parse. It then integrates the parse into the current context. Context consists of a set of frames and a set of global variables. As new extracted information arrives, it is put into the context frames and sometimes used to set global variables. The system provides a general-purpose library of routines for manipulating frames.

This "event driven" architecture functions similar to a production system. An incoming parse causes a set of actions to fire which modify the current context. After the parse has been integrated into the current context, the DM examines the context to decide what action to take next. The DM attempts the following actions in the order listed:

- Clarify if necessary
- Sign off if all done
- Retrieve data and present to user
- Prompt user for required information

The rules for deciding what to prompt for next are very straightforward. The frame in focus is set to be the frame produced in response to the user, or to the last system prompt.

- If there are unfilled required slots in the focus frame, then prompt for the highest priority unfilled slot in the frame.
- If there are no unfilled required slots in the focus frame, then prompt for the highest priority missing piece of information in the context.

Our mechanism does not have separate "user initiative" and "system initiative" modes. If the system has enough information to act on, then it does it. If it needs information, then it asks for it. The system does not require that the user respond to the prompt. The user can respond with anything and the system will parse the utterance and set the focus to the resulting frame. This allows the user to drive the dialog, but doesn't require it. The system prompts are organized locally, at the frame level. The dialog manager or user puts a frame in focus, and the system tries to fill it. This representation is easy to author, there is no separate dialog control specification required. It is also robust in that it has a simple control that has no state to lose track of.

An additional benefit of Dialog Manager mechanism is that it is very largely declarative. Most of the work done by a developer will be the creation of frames, forms and grammars. The system developer creates a task file that specifies the system ontology and templates for communicating about nodes in the hierarchy. The templates are filled in from the values in the frames to generate output in the desired language. This is the way we currently generate SQL queries and user prompts. An example task frame specification is:

Frame:Air
[Depart_Loc]+
Prompt: "where are you departing from"
[City_Name]*
Confirm: "You are departing from $([City_Name]).
Is that correct?"
Sql: "dep_$[leg_num] in (select airport_code from
airport_codes where city like '!%' $(and state_province
like '[Depart_Loc].[State]'))"
[Airport_Code]*

This example defines a frame with name Air and slot [Depart_Loc]. The child nodes of Depart_Loc are are [City_Name] and [Airport_Code]. The "+" after [Depart_Loc] indicates that it is a mandatory field. The Prompt string is the template for prompting for the node information. The "*" after [City_Name] and [Airport_Code] indicate that if either of them is filled, the parent node [Depart_Loc] is filled. The Confirm string is a template to prompt the user to confirm the values. The SQL string is the template to use the value in an SQL query to the database.

The system will prompt for all mandatory nodes that have prompts. Users may specify information in any order, but the system will prompt for whatever information is missing until the frame is complete.

1.7 Database & Internet Interface

The back-end interface consists of an SQL database and domain-specific Perl scripts for accessing information from the Internet. During operation, database requests are transmitted by the Dialog Manager to the database server via a formatted frame.

The back-end consists of a static and dynamic information component. Static tables contain data such as conversions between 3-letter airport codes and the city, state, and country of the airport (e.g., BOS for Boston Massachusetts). There are over 8000 airports in our database, 200 hotel chains, and 50 car rental companies. The dynamic information content consists of database tables for car, hotel, and airline flights.

When a database request is received, the Dialog Manager's SQL command is used to select records in local memory. If no records are found to match, the back-end can submit an HTTP-based request for the information via the Internet. Records returned from the Internet are then inserted as rows into the local SQL database and the SQL statement is once again applied.

1.8 Language Generation

The language generation module uses templates to generate text based on dialog speech acts. Example dialog acts include "prompt" for prompting the user for needed information, "summarize" for summarization of flights, hotels, and rental cars, and "clarify" for clarifying information such as departure and arrival cities that share the same name.

1.9 Text-to-Speech Synthesis

For audio output, we have developed a domain-dependent concatenative speech synthesizer. Our concatenative synthesizer can adjoin units ranging from phonemes, to words, to phrases and sentences. For domain modeling, we use a voice talent to record entire task-dependent utterances (e.g., "What are your travel plans?") as well as short phrases with carefully determined break points (e.g., "United flight", "ten", "thirty two", "departs Anchorage at"). Each utterance is orthographically transcribed and phonetically aligned using a HMM-based recognizer. Our research efforts for data collection are currently focused on methods for reducing the audible distortion at segment boundaries, optimization schemes for prompt generation, as well as tools for rapidly correcting boundary misalignments. In general, we find that some degree of hand-correction is always required in order to reduce distortions at concatenation points.

During synthesis, the text is automatically divided into individual sentences that are then synthesized and pipelined to the audio server. A text-to-phoneme conversion is applied using a phonetic dictionary. Words that do not appear in the phonetic dictionary are automatically pronounced using a multi-layer perceptron based pronunciation module. Here, a 5-letter context is extracted from the word to be pronounced. The letter input is fed through the MLP and a phonetic symbol (or possibly epsilon) is output by the network. By sliding the context window, we can extract the phonetic pronunciation of the word. The MLP is trained using letter-context and symbol output pairs from a large phonetic dictionary.

The selection of units to concatenate is determined using a hybrid search algorithm that operates at the word or phoneme level. During synthesis, sections of word-level text that have been recorded are automatically concatenated. Unrecorded words or word sequences are synthesized using a Viterbi beam search across all available phonetic units. The cost function includes information regarding phonetic context, pitch, duration, and signal amplitude. Audio segments making up the best-path are then concatenated to generate the final sentence waveform.

2. DATA COLLECTION & EVALUATION

2.1 Data Collection Efforts

Local Collection Effort

The Center for Spoken Language Research maintains a dialup Communicator system for data collection[1]. Users wishing to use the dialogue system can register at our web site [1] and receive a PIN code and system telephone number. To date, our system has fielded over 1750 calls totaling over 25,000 utterances from nearly 400 registered users.

NIST Multi-Site Data Collection

During the months of June and July of 2000, The National Institute of Standards (NIST) conducted a multi-site data collection effort for the nine DARPA Communicator participants. Participating sites included: AT&T, IBM, BBN, SRI, CMU, Colorado, MIT, Lucent, and MITRE. In this data collection, a pool of potential users was selected from various parts of the United States by a market research firm. The selected subjects were native speakers of American English who were possible frequent travelers. Users were asked to perform nine tasks. The first seven tasks consisted of fixed scenarios for one-way and round-trip flights both within and outside of the United States. The final two tasks consisted of users making open-ended business or vacation.

2.2 System Evaluation

Task Completion

A total of 72 calls from NIST participants were received by the CU Communicator system. Of these, 44 callers were female and 28 were male. Each scenario was inspected by hand and compared against the scenario provided by NIST to the subject. For the two open-ended tasks, judgment was made based on what the user asked for with that of the data provided to the user. In total, 53/72 (73.6%) of the tasks were completed successfully. A detailed error analysis can be found in [11].

Word Error Rate Analysis

A total of 1327 utterances were recorded from the 72 NIST calls. Of these, 1264 contained user speech. At the time of the June 2000 NIST evaluation, the CU Communicator system did not implement voice-based barge-in. We noticed that one source of error was due to users who spoke before the recording process was started. Even though a tone was presented to the user to signify the time to speak, 6.9% of the utterances contained instances in which the user spoke before the tone. Since all users were exposed to several other Communicator systems that

[2] The system can be accessed toll-free at 1-866-735-5189

employed voice barge-in, there may be some effect from exposure to those systems. Table 3 summarizes the word error rates for the system utilizing the June 2000 NIST data as the test set. Overall, the system had a word error rate (WER) of 26.0% when parallel gender-dependent decoders were utilized. Since June of 2000, we have collected an additional 15,000 task-dependent utterances. With the extra data, we were able to remove our dependence on the CMU Communicator training data [12]. When the language model was reestimated and language model weights reoptimized using only CU Communicator data, the WER dropped from 26.0% to 22.5%. This amounts to a 13.5% relative reduction in WER.

Table 1: CU Communicator Word Error Rates for (A) Speaker Independent acoustic models and June 2000 language model, (B) Gender-dependent parallel recognizers with June 2000 Language Model, and (C) Language Model retrained in December 2000.

June 2000 NIST Evaluation Data, 1264 utterances, 72 speakers	Word Error Rate
(A) Speaker Indep. HMMs (LM#1)	29.8%
(B) Gender Dependent HMMs (LM#1)	26.0%
(C) Gender Dependent HMMs (LM#2)	22.5%

Core Metrics

Sites in the DARPA Communicator program agreed to log a common set of metrics for their systems. The proposed set of metrics was: Task Completion, Time to Completion, Turns to Completion, User Words/Turn, System Words/Turn, User Concepts/Turn, Concept Efficiency, State of Itinerary, Error Messages, Help Messages, Response Latency, User Words to Completion, System Words to Completion, User Repeats, System Repeats/Reprompts, Word Error, Mean Length of System Utterance, and Mean Length of System Turn.

Table 2: Dialogue system evaluation metrics

Item	Min	Mean	Max
Time to Completion (secs)	120.9	260.3	537.2
Total Turns to Completion	23	37.6	61
Response Latency (secs)	1.5	1.9	2.4
User Words to Task End	19	39.4	105
System Words to End	173	331.9	914
Number of Reprompts	0	2.4	15

Table 2 summarizes results obtained from metrics derived automatically from the logged timing markers for the calls in which the user completed the task assigned to them. The average time to task completion is 260. During this period there are an average of 19 user turns and 19 computer turns (37.6 average total turns). The average response latency was 1.86 seconds. The response latency also includes the time required to access the data live from the Internet travel information provider.

3. CU MOVE

3.1 Task Overview

The "CU Move" system represents our work towards achieving graceful human-computer interaction in automobile environments. Initially, we have considered the task of vehicle route planning and navigation. As our work progresses, we will expand our dialog system to new tasks such as information retrieval and summarization and multimedia access.

The problem of voice dialog within vehicle environments offers some important speech research challenges. Speech recognition in car environments is in general fragile, with word-error-rates (WER) ranging from 30-65% depending on driving conditions. These changing environmental conditions include speaker changes (task stress, emotion, Lombard effect, etc.) as well as the acoustic environment (road/wind noise from windows, air conditioning, engine noise, exterior traffic, etc.).

In developing the CU-Move system [13,14], there are a number of research challenges that must be overcome to achieve reliable and natural voice interaction within the car environment. Since the speaker is performing a task (driving the vehicle), the driver will experience a measured level of user task stress and therefore this should be included in the speaker-modeling phase. Previous studies have clearly shown that the effects of speaker stress and Lombard effect can cause speech recognition systems to fail rapidly. In addition, microphone type and placement for in-vehicle speech collection can impact the level of acoustic background noise and speech recognition performance.

3.2 Signal Processing

Our research for robust recognition in automobile environments is concentrated on development of an intelligent microphone array. Here, we employ a Gaussian Mixture Model (GMM) based environmental classification scheme to characterize the noise conditions in the automobile. By integrating an environmental classification system into the microphone array design, decisions can be made as to how best to utilize a noise-adaptive frequency-partitioned iterative enhancement algorithm [15,16] or model-based adaptation algorithms [17,18] during decoding to optimize speech recognition accuracy on the beam-formed signal.

3.3 Data Collection

A five-channel microphone array was constructed using Knowles microphones and a multi-channel data recorder housing built (Fostex) for in-vehicle data collection. An additional reference microphone is situated behind the driver's seat. Fig. 3 shows the constructed microphone array and data recorder housing.

Figure 3: Microphone array and reference microphone (left), Fostex multi-channel data recorder (right).

As part of the CU-Move system formulation, a two phase data collection plan has been initiated. Phase I focuses on collecting acoustic noise and probe speech from a variety of cars and driving conditions. Phase II focuses on a extensive speaker collection across multiple U.S. sites. A total of eight vehicles have been selected for acoustic noise analysis. These include the

following: a compact car, minivan, cargo van, sport utility vehicle (SUV), compact and full size trucks, sports car, full size luxury car. A fixed 10 mile route through Boulder, CO was used for Phase I data collection. The route consisted of city (25 & 45mph) and highway driving (45 & 65mph). The route included stop-and-go traffic, and prescribed locations where driver/passenger windows, turn signals, wiper blades, air conditioning were operated. Each data collection run per car lasted approximately 35-45 minutes. A detailed acoustic analysis of Phase I data can be found in [13]. Our plan is to begin Phase II speech/dialogue data collection during spring 2001, which will include (i) phonetically balanced utterances, (ii) task-specific vocabularies, (iii) natural extemporaneous speech, and (iv) human-to-human and Wizard-of-Oz (WOZ) interaction with CU-Communicator and CU-Move dialog systems.

3.4 Prototype Dialog System

Finally, we have developed a prototype dialog system for data collection in the car environment. The dialog system is based on the MIT Galaxy-II Hub architecture with base system components derived from the CU Communicator system [1]. Users interacting with the dialog system can enter their origin and destination address by voice. Currently, 1107 street names for Boulder, CO area are modeled. The system can resolve street addresses by business name via interaction with an Internet telephone book. This allows users to ask more natural route queries (e.g., “I need an auto repair shop”, or “I need to get to the Boulder Marriott”). The dialog system automatically retrieves the driving instructions from the Internet using an online WWW route direction provider. Once downloaded, the driving directions are queried locally from an SQL database. During interaction, users mark their location on the route by providing spoken odometer readings. Odometer readings are needed since GPS information has not yet been integrated into the prototype dialog system. Given the odometer reading of the vehicle as an estimate of position, route information such as turn descriptions, distances, and summaries can be queried during travel (e.g., "What's my next turn", "How far is it", etc.).

The prototype system uses the CMU Sphinx-II speech recognizer with cellular telephone acoustic models along with the Phoenix Parser [10] for semantic parsing. The dialog manager is mixed-initiative and event driven. For route guidance, the natural language generator formats the driving instructions before presentation to the user by the text-to-speech server. For example, the direction, "Park Ave W. becomes 22nd St." is reformatted to, "Park Avenue West becomes Twenty Second Street". Here, knowledge of the task-domain can be used to significantly improve the quality of the output text. For speech synthesis, we have developed a Hub-compliant server that interfaces to the AT&T NextGen speech synthesizer.

3.5 Future Work

We have developed a Hub compliant server that interfaces a Garmin GPS-III global positioning device to a mobile computer via a serial port link. The GPS server reports vehicle velocity in the X,Y,Z directions as well as real-time updates of vehicle position in latitude and longitude. HRL Laboratories has developed a route server that interfaces to a major navigation content provider. The HRL route server can take GPS coordinates as inputs and can describe route maneuvers in terms of GPS coordinates. In the near-term, we will interface our GPS server to the HRL route server in order to provide real-time updating of vehicle position. This will eliminate the need for periodic location update by the user and also will allow for more interesting dialogs to be established (e.g., the computer might proactively tell the user about upcoming points of interest, etc.).

4. REFERENCES

[1] http://communicator.colorado.edu

[2] W. Ward, B. Pellom, "The CU Communicator System," *IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone Colorado, December, 1999.

[3] http://fofoca.mitre.org

[4] Seneff, S., Hurley, E., Lau, R., Pao, C., Schmid, P., Zue, V., “Galaxy-II: A Reference Architecture for Conversational System Development,” *Proc. ICSLP*, Sydney Australia, Vol. 3, pp. 931-934, 1998.

[5] Ravishankar, M.K., “Efficient Algorithms for Speech Recognition”. Unpublished Dissertation CMU-CS-96-138, Carnegie Mellon University, 1996

[6] K. Hacioglu, W. Ward, "Dialog-Context Dependent Language Modeling Using N-Grams and Stochastic Context-Free Grammars", *Proc. IEEE ICASSP*, Salt Lake City, May 2001.

[7] K. Hacioglu, W. Ward, "Combining Language Models : Oracle Approach", *Proc. Human Language Technology Conference*, San Diego, March 2001.

[8] R. San-Segundo, B. Pellom, W. Ward, J. M. Pardo, "Confidence Measures for Dialogue Management in the CU Communicator System," *Proc. IEEE ICASSP*, Istanbul Turkey, June 2000.

[9] R. San-Segundo, B. Pellom, K. Hacioglu, W. Ward, J.M. Pardo, "Confidence Measures for Dialogue Systems," *Proc. IEEE ICASSP*, Salt Lake City, May 2001.

[10] Ward, W., “Extracting Information From Spontaneous Speech”, *Proc. ICSLP*, September 1994.

[11] B. Pellom, W. Ward, S. Pradhan, "The CU Communicator: An Architecture for Dialogue Systems", *Proc. ICSLP*, Beijing China, November 2000.

[12] Eskenazi, M., Rudnicky, A., Gregory, K., Constantinides, P., Brennan, R., Bennett, K., Allen, J., “Data Collection and Processing in the Carnegie Mellon Communicator,” *Proc. Eurospeech-99*, Budapest, Hungary.

[13] J.H.L. Hansen, J. Plucienkowski, S. Gallant, B.L. Pellom, W. Ward, "CU-Move: Robust Speech Processing for In-Vehicle Speech Systems," *Proc. ICSLP*, vol. 1, pp. 524-527, Beijing, China, Oct. 2000.

[14] http://cumove.colorado.edu/

[15] J.H.L. Hansen, M.A. Clements, “Constrained Iterative Speech Enhancement with Application to Speech Recognition,” *IEEE Trans. Signal Proc.*, **39**(4):795-805, 1991.

[16] B. Pellom, J.H.L. Hansen, “An Improved Constrained Iterative Speech Enhancement Algorithm for Colored Noise Environments," IEEE Trans. Speech & Audio Proc., **6**(6):573-79, 1998.

[17] R. Sarikaya, J.H.L. Hansen, "Improved Jacobian Adaptation for Fast Acoustic Model Adaptation in Noisy Speech Recognition," *Proc. ICSLP*, vol. 3, pp. 702-705, Beijing, China, Oct. 2000.

[18] R. Sarikaya, J.H.L. Hansen, "PCA-PMC: A novel use of a priori knowledge for fast model combination," *Proc. ICASSP*, vol. II, pp. 1113-1116, Istanbul, Turkey, June 2000.