A proposal to develop domain and subtask-adaptive dialog management models

David Griol[1], *Zoraida Callejas*[2]

[1]Dept. of Computer Science, Carlos III University of Madrid, Spain
[2]Dept. of Languages and Computer Systems, University of Granada, Spain
dgriol@inf.uc3m.es, zoraida@ugr.es

10.21437/Interspeech.2015-458

Abstract

Statistical dialog management techniques have the main advantage of allowing an easy adaptation of the dialog model to different application domains. In this paper we propose to also adapt the operation of the dialog manager to the different subtasks that conform the structure of the dialog in each domain. To do so, we present a variation of a statistical dialog management technique to address this challenge. The evaluation results with a practical dialog system show that the use of such specific sub-task models increase the quality and number of successful interactions with the system.

Index Terms: Spoken Dialog Systems, Dialog Management, Dialog Structure, Dialog simulation.

1. Introduction

Dialog systems [1, 2, 3] go a step beyond traditional graphical user interfaces (GUIs) by adding the possibility to communicate with a machine through other interaction modes such as speech.

The performance of spoken language dialog systems has improved over time, extending initial application domains to more complex information retrieval and question answering applications [4], e-commerce systems [5], surveys applications [6], recommendations systems [7], e-learning and tutoring systems [8], in-car systems [9], remote control of devices and robots in smart environments [10], healthcare and Ambient Assisted Living systems [11], or embodied dialog systems and companions [12].

The described systems are usually designed ad-hoc for their specific domain using rule-based models and standards in which developers must specify the steps to be followed by the system. This way, the adaptation of the hand-crafted designed systems to consider specific users requirements or deal with new tasks is a time-consuming process that implies a considerable effort, with the ever-increasing problem of dialog complexity [13, 14].

In addition, although much work emphasizes the importance of taking into account user's and domain adapted models not only to solve the tasks presented to the dialog system by the user, but also to enhance the system performance in the communication task, this information is not usually considered when designing the dialog model for the system [15, 16]. For this reason, in most dialog applications, the dialog specification is the same for all cases: users typically have no control over the content or presentation of the service provided.

Incorporating intelligence into a spoken language based communication system requires, among other things, careful user modeling in conjunction with an effective dialog management. With the aim of creating dynamic and adapted dialogs, the application of statistical approaches to dialog modeling makes it possible to consider a wider space of dialog strategies in comparison to engineered rules [17, 18].

The main reason is that statistical approaches for dialog management can be trained from real dialogs, modeling the variability in user behaviors. Although the parameterization of the model depends on expert knowledge of the task, the final objective is to develop dialog systems that have a more robust behavior, better portability, and are easier to adapt to different user profiles or tasks [19].

In this paper we describe a framework to develop user and subtasks adapted spoken dialog systems. Our proposal is based on the definition of a statistical methodology for user modeling that estimates the user intention during the dialog and the current dialog subtask. These predictions, carried out for each user turn in the dialog, makes it possible to adapt the system dynamically. To do this, a statistical dialog model based on neural networks is generated taking into account these predictions and the history of the dialog up to the current moment. The next system response is selected by means of this model. The codification of the information and the definition of a data structure which takes into account the data supplied by the user throughout the dialog makes the estimation of the dialog model from the training data and practical domains manageable.

The remainder of the paper is organized as follows. Section 2 presents in detail our proposal to develop adaptive dialog systems. Section 3 describes the application of our proposal in the CMU Let's Go spoken dialog system, a system that has been used during the last years by the dialog systems community as a common ground for comparison and verifiable assessment of the improvements achieved. In Section 4 we discuss the evaluation results obtained in this application. Finally, in Section 5 we present the conclusions and outline guidelines for future work.

2. Proposed framework to develop adaptive spoken dialog systems

We represent dialogs as a sequence of pairs (A_i, U_i), where A_i is the output of the system (the system response or turn) at time i, and U_i is the semantic representation of the user turn (the result of the understanding process of the user input) at time i; both expressed in terms of dialog acts [18]. This way, each dialog is represented by:

$$(A_1, U_1), \cdots, (A_i, U_i), \cdots, (A_n, U_n)$$

where A_1 is the greeting turn of the system, and U_n is the last user turn. We refer to a pair (A_i, U_i) as S_i, the state of the dialog sequence at time i.

The lexical, syntactic and semantic information associated

to the speaker u's ith turn (U_i) is denoted as c_i^u. This information is usually represented by:

- the words uttered;
- part of speech tags, also called word classes or lexical categories. Common linguistic categories include noun, adjective, and verb, among others;
- predicate-argument structures, used by SLU modules in various contexts to represent relations within a sentence structure. They are usually represented as triples (subject-verb-object).
- named entities: sequences of words that refer to a unique identifier. This identifier may be a proper name (e.g., organization, person or location names), a time identifier (e.g., dates, time expressions or durations), or quantities and numerical expressions (e.g., monetary values, percentages or phone numbers).

2.1. User and subtask prediction models

Our model for task and subtask prediction is based on the one proposed in [20]. In this model, each user turn is modeled as a user action defined by a subtask to which the turn contributes, the dialog act of the turn, and its named entities. For example, for the Let's Go system, a subtask may be to provide the information necessary to perform a timetable query, the turn may be to provide the origin address, and the dialog act may be *Provide-Street*, being *Queen Avenue* the named entity involved.

For speaker u, DA_i^u denotes the dialog label of the ith turn, and ST_i^u denotes the subtask label to which the ith turn contributes. The dialog act is determined from the information about the turn and the previous dialog context (i.e., k previous utterances) as shown in Equation 1.

$$DA_i^u = \underset{d^u \in \mathcal{D}}{\operatorname{argmax}} P(d^u | c_i^u, ST_{i-1}^{i-k}, DA_{i-1}^{i-k}, c_{i-1}^{i-k}) \quad (1)$$

In a second stage, the subtask is determined from the lexical information, the dialog act computed according to Equation 1, and the dialog context, as shown in Equation 2.

$$ST_i^u = \underset{s^u \in \mathcal{S}}{\operatorname{argmax}} P(s^u | DA_i^u, c_i^u, ST_{i-1}^{i-k}, DA_{i-1}^{i-k}, c_{i-1}^{i-k}) \quad (2)$$

As described in [20], the conditional distributions shown in Equations 1 and 2 can be estimated by means of the general technique of choosing the maximum entropy (MaxEnt) distribution that properly estimates the average of each feature in the training data [21]. This can be written as a Gibbs distribution parameterized with weights λ as Equation 3 shows, where V is the size of the label set, X denotes the distribution of dialog acts or subtasks (DA_i^u or ST_i^u) and Φ denotes the vector of the described features used for user modeling.

$$P(X = st_i | \phi) = \frac{e^{\lambda_{st_i} \cdot \phi}}{\sum_{st=1}^{V} e^{\lambda_{st_i} \cdot \phi}} \quad (3)$$

Such calculation outperforms other state of the art approaches [20, 22, 21], as it increases the speed of training and makes possible to deal with large data sets. Each of the classes can be encoded as a bit vector such that, in the vector corresponding to each class, the ith bit is one and all other bits are zero. Then, V-one-versus-other binary classifiers are used as Equation 4 shows.

$$P(y|\phi) = 1 - P(\overline{y}|\phi) = \frac{e^{\lambda_y \cdot \phi}}{e^{\lambda_y \cdot \phi} + e^{\lambda_{\overline{y}} \cdot \phi}} = \frac{1}{1 + e^{-\lambda'_{\overline{y}} \cdot \phi}} \quad (4)$$

where $\lambda_{\overline{y}}$ is the parameter vector for the anti-label $\overline{y}$ and $\lambda'_{\overline{y}} = \lambda_y - \lambda_{\overline{y}}$.

2.2. Dialog Management

Considering the representation of dialogs as the sequence of pairs (A_i, U_i) described in the previous subsection, at time i, the objective of the dialog manager is to find the best system answer A_i. This selection is a local process for each time i and takes into account the previous history of the dialog [18]:

$$\hat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i | S_1, \cdots, S_{i-1}) \quad (5)$$

where set $\mathcal{A}$ contains all the possible system answers.

Following Equation 5, the dialog manager selects the next system response taking into account the sequence of previous pairs (A_i, U_i). The main problem to resolve this equation is usually the large number of possible sequences of states. To solve the problem, we define a data structure in order to establish a partition in this space, i.e., in the history of the dialog preceding time i). This data structure, which we call *Interaction Register* (IR), contains the following information:

- sequence of user dialog acts provided by the user throughout the previous history of the dialog (i.e., the output of the SLU module);
- predicted user dialog act (generated by Equation 1);
- predicted user subtask (generated by Equation 2).

After applying these considerations and establishing the equivalence relation in the histories of the dialogs, the selection of the best A_i is given by Equation 6.

$$\hat{A}_i = \underset{A_i \in \mathcal{A}}{\operatorname{argmax}} P(A_i | IR_{i-1}, S_{i-1}) \quad (6)$$

A user turn can also provide task-independent information (e.g., *Affirmation*, *Negation*, and *Not-Understood* dialog acts). This kind of information implies some decisions which are different from simply updating the IR_{i-1}. Hence, for the selection of the best system response A_i, we take into account the IR from turn 1 to turn $i-1$, and we explicitly consider the last state S_{i-1}.

For the dialog manager to determine the next system answer, we have assumed that the exact values of the task-dependent attributes are not significant. They are important for accessing data repositories and for constructing the output sentences of the system. However, the only information necessary to predict the next system action is the presence or absence of concepts and attributes (i.e. whether each relevant piece of information has been correctly provided or not). Therefore, the codification we use for this information in the IR is in terms of three values, $\{0, 1, 2\}$, according to the following criteria:

- (0) The concept is unknown or the value of the attribute is not given;
- (1) the concept or attribute is known with a confidence score that is higher than a given threshold.
- (2) the concept or attribute has a confidence score that is lower than the given threshold.

We propose to solve Equation 6 by approximating this equation by a learned function. To do this, every dialog situation is classified taking into account a set of classes $\mathcal{C}$, in which a class contains all the sequences that provide the same set of system actions (responses). The objective of the dialog manager at each moment is to select a class of this set $c \in \mathcal{C}$, so that the system answer is the one associated with the selected class. As described in [18], the classification function can be defined in several ways. The best results were obtained using a multilayer perceptron (MLP) where the input layer receives the current state of the dialog, which is represented by the term (IR_{i-1},A_i). The values of the output layer can be viewed as the a posteriori probability of selecting the different system responses given the current situation of the dialog.

Figure 1 summarizes the combination of the proposed user modeling and dialog management methodologies. As can be observed, the user modeling module provides a prediction of the next user dialog act and the current subtask of the dialog. The set of user dialog acts and predicted values for the current user's dialog act and subtask are used to update the interaction register. The dialog manager considers this register and the last system response for the selection of the next system action.

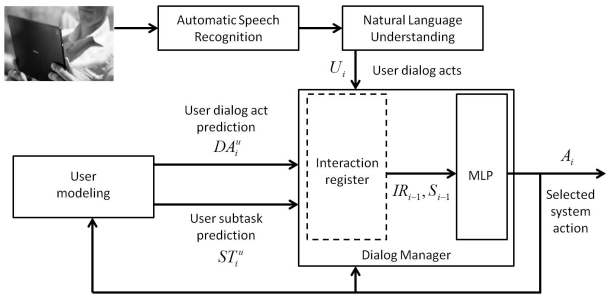

Figure 1: Combination of the proposed user modeling and dialog management methodologies for the development of adaptive spoken dialog systems

3. Application to the Let's Go dialog system

Let's Go is a spoken dialog system developed by the Carnegie Mellon University to provide bus schedule information in Pittsburgh [23]. In 2009, a corpus of 338 dialogs acquired with real users was distributed among the scientific community as a common testbed for the 2010 Spoken Dialog Challenge initiative [24]. The aim of the Challenge was to bring together multiple implementations of the same dialog task and deploy them in uncontrolled real user conditions and then make the results available for common evaluation techniques.

We have chosen the Let's Go task to evaluate our proposal for several reasons. Firstly, the corpus available was gathered from a real task in an operative dialog system that provided its service to real users. This poses a challenge to build realistic user models and find new dialog strategies that are at least as good as the hand-crafted system. Secondly, Let's Go is a common ground for experimentation and evaluation within the dialog system community, which therefore makes our results directly comparable to the alternatives presented by other authors, and this is why it has been intensively used by researchers in the last years [25, 24, 26, 27].

The 36 system dialog acts can also be classified into 5 groups: *formal* (dialog formalities like "welcome"), *results* (presentation of search results), *queries* (request for values to fill slots), *statusreports* (when the system reports about its status, e.g. "looking up database"), *error* (error messages), and *instructions* (instructions to the user how to speak to the system).

Category	Size	Example values
from	3	ftstop, ftmonument, ftneigh
fstop	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
fmon	52	"AIRPORT", "CENTURY SQUARE"
fneigh	220	"DOWNTOWN", "SQUIRREL HILL"
to	3	ttstop, ttmonument, ttneigh
tstop	328 455	"FORBES&MURRAY", "ANYWHERE&FORBES"
tmon	52	"AIRPORT", "CENTURY SQUARE"
tneigh	220	"DOWNTOWN", "SQUIRREL HILL"
time	2	"NEXT", time specific
hour	12	"ONE", "TWELVE"
min	60	"ZERO", "TEN", "THIRTY FIVE"
pd	2	"AM", "PM"
day	2	day 10, "TODAY", "WEDNESDAY"
tref	4	"ARRIVE BEFORE", "LEAVE AFTER"
meth	4	"RESTART", "FINISHED", constraints,
disc	9	"REPEAT", "FOLLOWING", "PREVIOUS", none

Table 1: User dialog acts categories defined in the Let's Go system [28]

The different objectives of the dialogs for the Spoken Dialog Challenge were labeled in the corpus by considering the different places and times for which the users required information (from one to five), users' requirements about previous and next buses, number of uncovered places, and possible system failures. The different combinations of these parameters in the corpus lead to the definition of 38 different objectives. The dialogs were also divided into 10 subtasks (*welcome*, *ask_for_query*, *ask_for_attribute*, *confirm_query*, *confirm_attribute*, *looking_up_database*, *provide_results*, *provide_instructions*, *query_error*, and *goodbye*).

A total number of 22 features define the pair (IR_{i-1}, S_{i-1}) for the Let's Go task: the last system response (A_{i-1}), 18 features corresponding to the Interaction Register IR_{i-1} (predicted current user dialog act, 16 task-dependent user dialog acts, and predicted current dialog subtask), and 3 features corresponding to the task-independent information (*Affirmation*, *Negation*, and *Not-Understood* dialog acts). This information is the input to our MLP and the number of features is similar to other application domains in which our proposal has been previously applied [18].

4. Evaluation

To assess the appropriateness of the dialog manager decisions taking into account also the result of the user and subtask models, we have firstly applied our proposal to learn these prediction models as explained in Subsection 2.1 using the initial Let's Go corpus of 338 dialogs. Then, this model was used to acquire a second corpus of 1,000 simulated dialogs by means of its interaction with an initial version of the Let's Go dialog system (DM_1).

Then, a 5-fold cross-validation process was used to carry out the evaluation. The initial corpus of 338 dialogs was randomly split into five subsets of 1,817 samples (20% of the corpus). Our experiment consisted of five trials. Each trial used a different subset taken from the five subsets as the test set, and the remaining 80% of the corpus was used as the training set

for the user and dialog models. A validation subset (20%) was extracted from each training set.

From our previous work on statistical dialog management [18], in this case we propose three measures to evaluate the quality of the responses selected by the statistical dialog manager. These measures are calculated by comparing the answer automatically generated by the statistical dialog manager (DM_2) for each input in the test partition with regard to the reference answer annotated in the corpus (DM_1). This way, the evaluation is carried out turn by turn. Thus, the aim is not to evaluate the complete dialog as a unit, but to assess the appropriateness of the dialog manager response for each sample in the test partition (i.e., current situations of the dialog). The three measures used for the described evaluation are:

- *Matching*: the percentage of responses provided by DM_2 that are equal to the reference answer in the corresponding turn of the test corpus;
- *Coherence*: the percentage of answers provided by DM_2 that are coherent with the current state of the dialog although they are not necessarily the same that the reference answer;
- *Error*: the percentage of answers provided by DM_2 that would cause the failure of the dialog.

The measure *Matching* is automatically calculated, evaluating whether the responses provided by DM_1 and DM_2 are the same. The calculation of the *Coherence* and *Error* measures requires expert annotation of the corpus. Thus, to decide about coherence of system responses, we asked three annotators to answer the following question: "Given the current dialog state: does it make sense that the system generates this response?". They were also advised about considering user's adaptation as an important criterion to answer the question. The responses labeled as *Error* correspond to those that have not been considered coherent.

Table 2 shows the results of the proposed evaluation that compared the initial dialog manager developed for the Let's Go system (DM_1) and the dialog manager developed using our complete proposal (DM_2). The values obtained for the matching and coherence measures show that the DM_2 dialog manager deviates from the initial dialog model and provides new valid paths to achieve each one of the required objectives defined in each task. This way, exact matches between DM_1 and DM_2 were reduced while coherence increased, as most of the non-matching responses were coherent and thus acceptable for the task.

A deeper study of the system responses provided by both dialog managers showed that DM_2 by considering the information provided by the user model was able to tackle new situations and generate new coherent answers for the situations already present in the initial corpus. Also it could avoid previously detected errors anticipating the user's intention and was better prepared for future user's actions being able to disambiguate between different alternatives for the user's dialog acts at each turn.

Moreover, the codification developed to represent the state of the dialog and the good operation of the MLP classifier make it possible for the number of responses that cause the failure of the system to be only 2.86% for the DM_2 dialog manager, instead of the initial 5.48% in DM_1.

With respect to the dialog style features, we measured the balance between different types of system dialog acts using DM_1 and DM_2. The results, showed in Table 3, indicate that

	Matching	*Coherence*	*Error*
DM_1	93.28%	94.52%	5.48%
DM_2	81.22%	97.14%	2.86%

Table 2: Evaluation results obtained with the DMs developed for the Let's Go system

using DM_2 there was an increment in the number of system turns that actually provide information to the user, which is consistent with the fact that the task completion rate is higher using our dialog manager.

	DM_1	DM_2
S_Confirm	39.16	36.19
S_Request	20.08	18.21
S_Inform	40.39	45.37
S_Other	0.37	0.23

Table 3: Percentages of system dialog acts using DM_1 and DM_2

In addition, we grouped all user and system dialog acts into "goal directed" (actions to provide or request information) and "grounding" actions (dialog formalities, unrecognized actions, confirmations, and negations). The results in Table 4 show that the dialogs acquired with DM_2 are better as the proportion of goal-directed actions increases for this system.

	DM_1	DM_2
Goal-directed actions	73.16	78.89
Grounding actions	26.84	21.11

Table 4: Percentages of goal directed and grounding actions using DM_1 and DM_2

5. Conclusions

In this paper, we contribute a framework which can be used to develop adaptive spoken dialog systems. Our proposal is based on the definition of a statistical methodology for user and task modeling that anticipates the next user turn during the dialog and makes it possible to adapt the system dynamically to the user's needs. To do this, a statistical dialog model based on neural networks selects the next system response taking into account the prediction of the user's intention and the history of the dialog up to the current dialog state.

We have provided a complete implementation of our framework for the Let's Go dialog system. The results of the evaluation show that the number of coherent responses provided by the statistical dialog manager increases with respect to the baseline, while the number of responses that lead to dialog failure decreases. The dialog manager also improves the confirmation and error correction rates for the different tasks.

For future work we plan to apply the proposed technique to other tasks in order to see whether it can be used for comparison between several adaptation models and dialog management techniques. We also intend to extend the evaluation of the system considering user satisfaction measures that complement the statistical measures employed.

6. References

[1] M. McTear and Z. Callejas, *Voice Application Development for Android*. Packt Publishing, 2013.

[2] R. Pieraccini, *The Voice in the Machine: Building computers that understand speech*. MIT Press, 2012.

[3] M. F. McTear, *Spoken Dialogue Technology: Towards the Conversational User Interface*. Springer, 2004.

[4] F. Metze, X. Anguera, E. Barnard, M. Davel, and G. Gravier, "Language independent search in MediaEval's Spoken Web Search task," *Computer, Speech and Language*, vol. 28(5), pp. 1066–1082, 2014.

[5] M. Tsai, "The VoiceXML dialog system for the e-commerce ordering service," in *Proc. CSCWD*, 2005, pp. 95–100.

[6] A. Stent, S. Stenchikova, and M. Marge, "Reinforcement learning of dialogue strategies with hierarchical abstract machines," in *Proc. SLT*, 2006, pp. 210–213.

[7] J. Chai, V. Horvath, N. Nicolov, M. Stys, N. Kambhatla, W. Zadrozny, and P. Melville, "Natural Language Assistant: A Dialog System for Online Product Recommendation," *AI Magazine*, vol. 23, pp. 63–75, 2002.

[8] K. Kopp, M. Britt, K. Millis, and A. Graesser, "Improving the efficiency of dialogue in tutoring," *Learning and Instruction*, vol. 22(5), pp. 320–330, 2012.

[9] H. Hofmann, A. Silberstein, U. Ehrlich, A. Berton, C. Muller, and A. Mahr, *Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialog Systems into Practice*. Springer Science+Business Media, 2014, ch. Development of Speech-Based In-Car HMI Concepts for Information Exchange Internet Apps, pp. 15–28.

[10] G. Skantze, A. Hjalmarsson, and C. Oertel, "Turn-taking, feedback and joint attention in situated human-robot interaction," *Speech Communication*, vol. 65, pp. 50–66, 2014.

[11] T. Bickmore, K. Puskar, E. Schlenk, L. Pfeifer, and S. Sereika, "Maintaining reality: Relational agents for antipsychotic medication adherence," *Interacting with Computers*, vol. 22, pp. 276–288, 2010.

[12] O. Horchak, J.-C. Giger, M. Cabral, and G. Pochwatko, "From demonstration to theory in embodied language comprehension: A review," *Cognitive Systems Research*, vol. 29-30, pp. 66–85, 2014.

[13] T. Paek and R. Pieraccini, "Automating spoken dialogue management design using machine learning: An industry perspective," *Speech Communication*, vol. 50(8-9), pp. 716–729, 2008.

[14] J. Rouillard, "Web services and speech-based applications around VoiceXML," *Journal of Networks*, vol. 2(1), pp. 27–35, 2007.

[15] S. Seneff, M. Adler, J. Glass, B. Sherry, T. Hazen, C. Wang, and T. Wu, "Exploiting Context Information in Spoken Dialogue Interaction with Mobile Devices," in *Proc. IMUx*, 2007, pp. 1–11.

[16] S. Kartakis, "A Design-and-Play Approach to Accesible User Interface Development in Ambient Intelligence Environments," *Journal Computers in Industry*, vol. 61(4), pp. 318–328, 2010.

[17] S. Young, "The Statistical Approach to the Design of Spoken Dialogue Systems," Cambridge University Engineering Department (UK), Tech. Rep., 2002.

[18] D. Griol, Z. Callejas, R. López-Cózar, and G. Riccardi, "A domain-independent statistical methodology for dialog management in spoken dialog systems," *Computer Speech and Language*, vol. 28, no. 3, pp. 743–768, 2014.

[19] J. Schatzmann, K. Weilhammer, M. Stuttle, and S. Young, "A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies," *Knowledge Engineering Review*, vol. 21(2), pp. 97–126, 2006.

[20] S. Bangalore, G. DiFabbrizio, and A. Stent, "Learning the Structure of Task-Driven Human-Human Dialogs," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16(7), pp. 1249–1259, 2008.

[21] A. Berger, S. Pietra, and V. Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22(1), pp. 39–71, 1996.

[22] P. Haffner, "Scaling large margin classifiers for spoken language understanding," *Speech Communication*, vol. 48(4), pp. 239–261, 2006.

[23] A. Raux, B. Langner, A. Black, and M. Eskenazi, "Let's go public! taking a spoken dialog system to the real world," in *Proc. Interspeech*, 2005, pp. 885–888.

[24] A. Black, S. Burger, B. Langner, G. Parent, and M. Eskenazi, "Spoken dialog challenge 2010," in *Proc. IEEE SLT*, 2010, pp. 448–453.

[25] A. Schmitt, S. Ultes, and W. Minker, "A Parameterized and Annotated Spoken Dialog Corpus of the CMU Let's Go Bus Information System," in *Proc. LREC*, 2012, pp. 3369–3375.

[26] J. Williams, I. Arizmendi, and A. Conkie, "Demonstration of AT&T Let's Go: A production-grade statistical spoken dialog system," in *Proc. SLT*, 2010, pp. 157–158.

[27] H. Hastie, N. Merigaud, X. Liu, and O. Lemon, ""Let's Go, DUDE!" Using the Spoken Dialogue Challenge to teach Spoken Dialogue development," in *Proc. IEEE SLT*, 2010, pp. 466–471.

[28] B. Thomson, K. Yu, S. Keizer, M. Gasic, F. Jurcicek, F. Mairesse, and S. Young, "Bayesian dialogue system for the Let's Go Spoken Dialogue Challenge," in *Proc. IEEE SLT*, 2010, pp. 460–465.