RC22779 (W0304-149) April 15, 2003
Computer Science

IBM Research Report

Intelligent Information Navigation in Conversation Systems

Shimei Pan, Rosario A. Uceda-Sosa
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Intelligent Information Navigation in Conversation Systems

Abstract

We define *information navigation* as strategies used by a conversation system to help a user locate targets in an information space. *Intelligent information navigation* concerns how a system can most effectively guide a user to find his targets. It is useful in information seeking applications such as finding flight or house information from databases. Unlike the static approaches adopted in today's conversation systems, we propose an approach that is sensitive to the data in the current search space as well as the user's search preferences. Our results demonstrate the effectiveness and feasibility of this approach.

1 Introduction

Information navigation concerns how a system acquires information to form a data query. This includes the order in which information should be acquired as well as whether additional information should be acquired. For example, in order to form a query to find house information, a system needs to acquire many different information such as *price*, *town*, *school*, *bedroom number*, *bathroom number* and various amenities[1]. To achieve this, the system can either rely on the user to provide information voluntarily or actively guide the user following a productive path. For the user-guided navigation approach to work well, since the system operates in the user initiative mode, both the speech recognizer and the natural language interpreter have to work well so that the system understands flexible and complex user queries. Moreover, it also requires the user to understand the system's capability so that he knows when and what to ask at each conversation state. In addition, the user also needs to know the data in the current search space so that he has enough knowledge to choose optimal navigation paths. For example, *house style* is not a good navigation feature because the system will return either the previous results or empty set when all the houses in the current search space are "colonial". All these requirements pose challenges to today's conversation systems and make the user-guided approach inadequate in information navigation. Thus, when needed, the conversation system should actively provide help and guide the user in information seeking.

In many existing conversation systems, if multiple attributes are missing, which attribute should be prompted next is pre-determined. For example, an application designer may decide that attribute *A* is more important than attribute *B*. Thus, the system should always acquire A before B if both A and B are unspecified. Because the relative order in which the system acquires *A* and *B* will not change from user to user and from conversation to conversation, we call it a *static navigation approach*. Moreover, before it issues a data query, the system will keep acquiring missing information until all of them are obtained.

There are problems with the static navigation strategy. First, if there are too many features to be acquired, prompting each unspecified feature one by

[1] None of the above information is mandatory because a user can use any features to find his preferred houses.

one until all the features are acquired is likely to produce a long and tedious conversation. For example, in a real estate application, a house may have a few dozen features. Acquiring all of them one by one will take many turns. An intelligent system needs to know when to stop if obtaining more constraint is unnecessary. Second, this strategy tends to produce over-constrained queries. This occurs because the system acquires new constraints in a pre-determined manner, regardless of what is in the search space. Over-constrained query results are empty and thus, not useful to a user. Third, since the order in which a system prompts for missing information is predetermined, it will not change from user to user. This one-size-fits-all strategy may not be the best for each individual user. To overcome these problems, we propose an *efficiency-based dynamic data navigation approach* that is sensitive to the data in the search space. In addition, we also incorporate a user's preferences so that the navigation is sensitive to a user's specific needs.

Information navigation is only one functionality implemented in a multimodal conversation system that helps a user find houses in a given region. A user interacts with the system using multiple input channels such as speech and gesture. The system acts/reacts to a user's request/response with automatically generated speech and graphics presentation. In addition to information navigation, a conversation manager has many other responsibilities such as generating a set of conversation acts based on a user's input and a current conversation context; tracking initiatives; handling exceptions like ambiguity, timeout, unknown input, and incomplete input; retrieving and manipulating information from the back channel, as well as constructing and maintaining the conversation history. Since we can not address all these topics in one paper, we only focus on the issues in intelligent information navigation.

2 Related Work

A large number of conversation systems have been developed to provide access to information systems such as travel information (Pellom et al., 2001; Potamianos et al., 2000; Litman and Pan, 1999; Rudnicky et al., 1999; Carlson, 1996; Albesano et al., 1997), movie information (Chu-Carroll, 2000), automobile information (Goddeau et al., 1996), financial information (Papineni et al., 1999) and weather forecast (Zue et al., 2000). However, there has not been much effort known to us that focuses on intelligent information navigation. Most systems still use static navigation. For example, in (Litman and Pan, 1999), a telephone-based train schedule system is implemented using a Finite-State-Machine (FSM)-based dialog manager. It handles dialogue management by explicitly enumerating all possible dialogue states, as well as allowable transitions between states. Before it queries the web for train information, the system needs to acquire four mandatory features: departure city, arrival city, departure data and departure time. Since the system allows mixed-initiative conversations, the user can specify any constraints in the order he chooses. However, if there is any missing information, the system uses static information navigation and prompts each value one by one using an order determined at the system design stage.

Form-based conversation management is another approach commonly used in today's conversation systems (D, 1999; Papineni et al., 1999; Goddeau et al., 1996). For example, in (Goddeau et al., 1996), the *Wheels* conversation system can help use find automobile information. The conversation is modeled based on the notion of filling in a form consisting of slots like *make*, *model*, *price*, *mileage* and *year* etc. The user again can provide any information in an order he chooses. However, if after the user specifies his constraints, there are still too many cars in the database, the system will cycle through an ordered list of prompts, choosing the first one whose corresponding field in the form remains empty. Thus, in terms of system-guided information navigation, the Wheels system still uses a static approach.

Unlike the approaches described in this section, we propose a dynamic information navigation approach that is sensitive to the changes in different conversation states. The order in which information is acquired will be different from one conversation to another. In addition, we also incorporate user preferences in navigation so that the search is specifically tailored to each user's special needs.

3 Efficiency-based Dynamic Navigation

Efficiency-based navigation is designed to help a user find his target quickly. In this paper, efficiency is defined as the number of system or user turns needed to reach a target, assuming the system only prompts the value of one new attribute at a time. Efficiency-based navigation minimizes the number of exchanges required to reach a target. To apply this strategy, at a given point of a conversation, the system uses greedy search to select, among all the unspecified features, a *navigation feature*. The navigation feature, if prompted first, is likely to minimize the number of turns needed to reach a target. In the following, first we describe how to use greedy search to find the next navigation feature. Then we describe the experiments that demonstrate the effectiveness of this approach.

3.1 The Approach

To select a navigation feature, our system computes the *expected search space reduction* for each unspecified feature. The one with the largest reduction is chosen as the navigation feature. The expected search space reduction for a categorical feature f_i is defined as:

$$R_{f_i} = 1 - \Sigma_x Prob(x) \times Prob(y|y = x) \quad (1)$$

Where $Prob(x)$ is the likelihood a user selects x when he is prompted for the value of f_i and $Prob(y|y = x)$ is the probability distribution of x in the current search space. Since the real $Prob(x)$ is sensitive to a user's preferences, R_{f_i} is sensitive to a user's special needs. Moreover, since $Prob(y|y = x)$ is sensitive to the data in the current search space, R_{f-i} changes dynamically when each new query constraint is acquired. Because $Prob(y|y = x)$ is also a measure of the size of the resulting search space, R_{f_i} directly measures the excepted size reduction if f_i is prompted next.

For continuous attributes like the *house price* or *size*, people usually care more about a range with a *max* and *min* than a specific value. Thus, two possible navigation features, g_{i-max} and g_{i-min}, are used for each continuous attribute g_i. Their expected size reduction can be computed as:

$$R_{g_{i-max}} = 1 - \int_{-\infty}^{\infty} Prob(x) \times Prob(y|y < x) \quad (2)$$

$$R_{g_{i-min}} = 1 - \int_{-\infty}^{\infty} Prob(x) \times Prob(y|y > x) \quad (3)$$

Where $Prob(x)$ is the probability a user selects x when the system prompts for g_{i-max} (or g_{i-min}), and $Prob(y|y < x)$ (or $Prob(y|y > x)$) is the probability of y satisfying $(y < x)$ (or $(y > x)$) in the current search space. Similar to R_{f_i}, both $R_{g_{i-max}}$ and $R_{g_{i-min}}$ are sensitive to user preferences as well as data in the search space. Finally, the feature that maximizes R_{f_i}, $R_{g_{i-max}}$ and $R_{g_{i-min}}$ is selected as the navigation feature and will be prompted next. In the following, we use experiments to demonstrate how each probability distribution is estimated. We also report the experiment results that confirm the effectiveness of this approach.

3.1.1 The Experiments

To demonstrate the effectiveness of this approach, we apply it to a real estate application in which 18 possible navigation features are used[2]. We use conversation simulations instead of full-fledged conversation systems for two reasons. First, many factors are known to affect a system's performance, such as efficiency (Walker et al., 1997). It is hard to tear these factors apart in a real human-computer conversation. Since we are only concerned about how the conversation efficiency is affected by different information navigation approaches, it is easier to use simulation to control the experiments. Second, without involving real human subjects, we can easily repeat the experiment over a large data set, which may not be possible with real human subjects. For example, it will require hundreds of conversation sessions before the same statistics can be computed.

The data in the experiment are house information taken from the *Multiple Listing Service (MLS)* database. There are a total of 325 houses in the testing database. The experiment starts with the system randomly picking a target house from the database. Then it initiates a conversation to locate the target house. At each point of the conversation, the system computes the expected space reduction for each unspecified feature. The feature with the largest space reduction is chosen as the navigation feature. Once a

[2]We did not use all the attributes in the MLS database because some of the fields are not clean. For example, the value of *basement* contains free text. Without any clean up, we can not use it in a meaningful way

navigation feature is prompted, the value of the target house is used as the answer. The search starts with the entire database. Each time a new constraint is applied, the search space is reduced. The search stops when the number of houses in the search space is small enough to be presented directly[3]. We repeatedly use a different house as the target and apply the same process until all the 325 houses in the database are tested. This experiment simulates situations in which a user knows exactly what he wants and the system tries to help him find his target house quickly.

To compute the expected search space reduction, we need to estimate two probabilities: the probability of a user choosing a particular value x when a navigation feature is prompted ($Prob(x)$) and the probability distribution of value x in the current search space. The second probability is relatively easy to compute. For categorical variables, $Prob(y|y = x) = \frac{|y=x|}{|x|}$, where $|y = x|$ is the number of entities satisfying this constraint and $|x|$ is the number of entities in the current search space. Similarly, for continuous variable g_{i-max}, $Prob(y|y < x) = \frac{|y<x|}{|x|}$, and for g_{i-min}, $Prob(y|y > x) = \frac{|y>x|}{|x|}$.

To estimate $Prob(x)$, which is how likely a user is to choose x when a navigation feature is prompted, we start with a simple case in which a particular user's preferences are not taken into consideration. In this case, we estimate $Prob(x)$ using the preferences of the general population. Since a region-wise poll on home buyer's preferences is hard to conduct, we approximate this using the distribution of the houses in the entire region by assuming the house supply and demand in an area are balanced. Because of this, $Prob(x)$ for a categorical variable f_i is estimated as the probability distribution of each x in the entire MLS database. For a continuous variable g_i, $Prob(x)$ in both g_{i-max} and g_{i-min} can be estimated using a normal distribution with mean μ equal to the average of g_i and standard deviation σ equal to $\sqrt{\frac{\Sigma(x-\mu)^2}{n}}$, where n is the number of houses in the database.

Since we did not take a particular user's preferences into consideration in this experiment, the probability for a user to choose x when a navigation feature is prompted is a static distribution. It will not change from one user to another. It is only affected by the preferences of the general population. However, the probability distribution of each x in the current search space will dynamically change from one conversation state to another.

Table 1: *Efficiency-based Navigation: an Example*

Nav. Fea.	Exp. Red.	Tgt. Val.	Sp. Size
heat	0.7633	Hot Air	325 to 38
garage	0.6500	3	38 to 7
bed(max)	0.9532	7	7 to 7
bath(max)	0.9448	8.2	7 to 7
price(max)	0.6633	5999000	7 to 7
tax (max)	0.6458	1100	7 to 1

Table 1 shows how the system locates a target house. At each navigation step, we list the navigation feature selected by the system and the expected search space reduction if the navigation feature is prompted next. We also list the target value derived from the target house and the size reduction when the new constraint is applied. At the very beginning, *heat* is chosen as the navigation feature. Its expected search space reduction is 76.33%. When *heat* is prompted, the value of the target house, *hot air*, is used as the answer. Once this new constraint is applied, the search space is reduced from 325 to 38. In the next step, since the search space has changed, the system re-computes the expected search space reduction for each unused feature. This time, *garage* has the largest value with expected space reduction equals to 65%. When *garage* is used as the navigation feature, the search space is further reduced to 7. This process is repeated until the target is located. Overall, only 6 out of the 18 possible navigation features are used in locating the target. The final statistics in table 2 include the navigation approaches used, the average number of exchanges needed to reach a target. We compare this approach with a static feature selection approach in which each feature is prompted one by one until all the features are used. The order in which each feature is acquired in the static approach is also fixed[4]. Finally, we compute the statistical significancy to verify whether

[3]Right now, we stop the search if less than three houses are in the search space.

[4]The order used in this experiment is randomly determined.

Table 2: *Results on Navigation Efficiency*

Approach	Average Exchanges	Significance
Dynamic	4.91	0.01
Static	18	na

there is any improvement in conversation efficiency when the new approach is used. The results in table 2 indicate that we are able to significantly improve the conversation efficiency by employing an efficiency-based approach that is sensitive to the dynamically changing search space. The average number of exchanges require to reach a target is reduced from 18, as in the static approach, to 4.91, as in the efficiency-based approach. Based on the one-sample t test, this difference is statistically significant with $\rho < 0.01$. This experiment confirms our assumption that efficiency-based dynamic navigation can help a user reach his target faster than the static one. In the following, we demonstrate how user preferences can be used to facilitate information navigation.

4 Combining Conversation Efficiency with User Preferences

Efficiency-based navigation is designed to help a user find his target quickly. In the previous section, we did not take a user's preferences into consideration. In this section, we will show that we are not only able to further improve conversation efficiency but also guide the search towards good matches when there is no exact match in the database for a target house. We do this by incorporating user preferences.

4.1 User Preferences

We record two types of preferences in our user model. First, which attributes are more important to a user. Second, for each feature, important or unimportant, what are the preferred values. For example, in the real estate application, a user may consider *price* and *location* the most important features. In addition, the user also prefers that the *house price* is between 300 and 600 thousand dollars and he prefers to live in the town of *A* or *B*.

User preferences affect information navigation in two aspects. First, instead of using the parameters estimated using the preferences of the general population, we estimate $Prob(x)$ more accurately using a user's own preferences. Since the computation is tailored to an individual user's needs, the efficiency-based strategy may work better for each user. Second, when no exact match can be found, we expect the system to match features that are most important to the user first.

4.2 The Combined Measure

The new preference-based navigation is based on a weighted combination of two factors: the expected search space reduction R_{f_i} and feature importance I_{f_i}. The definition for R_{f_i} is the same as that in section 3.1. The feature importance I_{f_i} is a binary variable. It is *1* when the selected feature is an important feature and *0* if the feature is unimportant. The weight β is also a number between *1* and *0*. It is used to either direct the navigation towards finding a target fast or matching as many important attributes as possible. The combine navigation measure C_{f_i} is defined as:

$$C_{f_i} = \beta * R_{f_i} + (1 - \beta) * I_{f_i} \quad (4)$$

Moreover, $Prob(x)$ in R_{f_i} or R_{g_i} has to be revised based on a user's preferences. To revise $Prob(x)$ for a categorical variable f_i, first we split the entire space into S_{i-in} and S_{i-out}. S_{i-in} contains values preferred by the user and S_{i-out} contains the rest. Since the user prefers the values in S_{i-in}, the probability of the user selecting a value from S_{i-in} should increase. In contrast, the probability for the values in S_{i-out} should decrease. More precisely, the revised probability is computed as:

$$Prob\prime(x \in S_{i-in}) = Prob(x) + \frac{(1-\alpha)\Sigma_{y \in S_{i-out}} Prob(y)}{|x|} \quad (5)$$

$$Prob\prime(y \in S_{i-out}) = \alpha \times Prob(y) \quad (6)$$

where x represents values in S_{i-in} and y are values in S_{i-out}, $Prob(x)$ and $Prob(y)$ are the original probability estimated from the general population, $Prob\prime(x)$ and $Prob\prime(y)$ are the revised probability estimated based on a user's preference, $|x|$ is the number of values in S_{i-in}, and α is a discount factor, which is a value between *1* and *0*. If $\alpha = 1$, the revised probability is the same as the original.

If $\alpha = 0$, the user will never chooses values from S_{i-out}

To revise $Prob(x)$ for a continuous variable g_i, the new probability distribution is a normal distribution with parameters derived from user preferences. Assume the preferred range for x is (min, max). Instead of estimating μ and σ using the preferences of the general population, the new μ for g_{i-max} is the max specified by the user. The new μ for g_{i-min} is min. The new σ is $\gamma \times (max - min)$ where γ is a flexibility factor. The larger γ is, the more flexible a user's constraint is. For example, if a user's preferred *price range* is from \$100K to \$200K, and $\gamma = 0.2$, the probability distribution for a user to choose x as the max is a normal distribution with μ equals to \$200k and σ equals to \$20k. Similarly, the probability for a user to choose x as the min is a normal distribution with μ equals to \$100k and σ equals to \$20k.

4.3 The Experiments

In this section, we use two experiments to demonstrate the usefulness of user preferences in information navigation. In the first experiment, we want to show that the number of exchanges required to reach a target can be further reduced because of more accurate $Prob(x)$. In the second experiment, we take both conversation efficiency and closeness to the target into consideration.

4.3.1 Revised Efficiency-based Navigation Using Preferences

Before the start of the first experiment, we create a user model in which each feature is assigned a few preferred values. We also select a few important features. Table 3 shows the user model where "NP" means "no preference". Features with * are considered important.

In this experiment, the searching targets are houses satisfying the user model. So, the set of target houses is a subset of the entire house set. There are a total of 68 target houses retrieved based on the user model. For each target house, the system starts a conversation to locate it. At each step, the system selects a navigation feature based on the revised $Prob\prime(x)$. Since in each simulated conversation session, the system only locates a specific target house with $(min == max)$ for all the continuous variable, we re-adjust the μ for g_{i-max} and g_{i-min} so that both of them are equal to $\frac{max+min}{2}$, where max and min are the preferred values specified in the user model. When the value of the navigation feature is prompted, the corresponding value in the target house is provided. The process is repeated for all the target houses. This experiment simulates the conversation behavior when user preferences are used. Our target houses are houses satisfying user preferences. Locating the target houses simulates a user searching preferred houses.

Table 3: *User Profile Used in the Experiment*

Feature	Prefer Value
Style	Tudor, Colonial
$Price_{max}*$	\$5M
$Price_{min}*$	\$4M
Tax_{min}	*NP*
Tax_{max}	*NP*
$Size_{max}$	*NP*
$Size_{min}$	*NP*
$BuiltYear_{min}$	*NP*
$BuiltYear_{max}$	*NP*
$bedroom_{max}$	9
$bedroom_{min}$	4
$bath_{max}$	9.0
$bath_{min}$	3.1
garage	NP
*Town**	$Town_s, Town_b$
water	*NP*
sewer	*NP*
heating	*NP*

Table 4 shows how the same house described in section 3.1.1 is retrieved here. In this process, the system uses a different probability estimation $Prob\prime(x)$ that is sensitive to a user's preferences. Based on the new probability, the expected space reduction for *bath(min)* is 89.22%, which is higher than 76.33%, the expected reduction for *heat*. As a result, *bath(min)* is prompted first. In this case, the minimum number of bathroom constraint is so good that the system located the same target house in one step.

The final results shown in table 5 include the average number of exchanges used to reach a target. We compare this performance with the one without

Table 4: *Efficiency-based Navigation with Preferences: an Example*

Nav. Fea.	Exp. Red.	Tgt. Val.	Sp. Size
bath(min)	0.8922	8.2	325 to 1

Table 5: *Navigation Results with or without User Preferences*

Approach	Ave. Exchanges	Significance
w/o preference	4.91	0.01
preference	4.10	na

user preferences in which $Prob(x)$ is estimated using the preference of the general population. The comparison results indicate that the system in average uses 4.10 exchanges to reach a target when user preferences are used. This is less than the 4.91 steps needed in the previous experiment. The difference is statistically significant with $\rho < 0.01$ based on the two-sample *t test*. These results confirm that using more accurate preference-based probability estimation further improves conversation efficiency.

4.3.2 Combining Efficiency with Importance

Based on our previous experiment, user preferences indeed improve navigation efficiency. But the experiment only simulates situations where a target house exists in the database. In reality, there may not exist a house in the database that matches all the preferences. That's where *feature importance* I_{f_i} can play a role. In the following experiment, we use the combined measure C_{f_i} describe in section 4.2 as the navigation criteria.

In this experiment, we randomly generate target houses that satisfy the user model. Since the houses are randomly generated, some combinations do not exist in the database. In fact, the values generated for the continuous variables are so specific, almost none of the target houses are in the database. This experiment is used to demonstrate the combined effects of navigation efficiency and feature importance. There are a total of 50 houses generated. For each target house, the system starts a conversation sequence to locate it. The conversation stops when the number of houses in the search space is small enough to be presented directly. After the same process is repeated for all the target houses, a set of statistics is computed, including how fast the system returns results and the similarity between the retrieved and the target houses. The similarity between a retrieved house X and a target house T, S(X,T), is defined as:

$$S(X,T) = \Sigma_{fi} Sim(X_{f_i}, T_{f_i}) \tag{7}$$

where

$$\begin{aligned} &Sim(X_{f_i}, T_{f_i}) = \\ &\quad 2 \leftarrow (X_{f_i} = T_{f_i}) \wedge (f_i \in Important) \\ &\quad 1 \leftarrow (X_{f_i} = T_{f_i}) \wedge (f_i \in Unimportant) \\ &\quad 0 \leftarrow (X_{f_i} \neq T_{f_i}) \end{aligned} \tag{8}$$

Based on equation (7), the similarity between a retrieved house X and a target house T is the sum of the similarity of each individual feature. The similarity of a single feature shown in equation (8) has three possible values: *2* if an important feature matches, *1* if an unimportant feature matches, and *0* if a feature does not match. Thus, the overall similarity score $S(X,T)$ can be used to evaluate the quality of the retrieved houses. In general the higher the score is, the closer the houses are to the user's target. If multiple houses are retrieved for a single target house, we use the average similarity score in the result. We run the experiment using C_{f_i} as the search criteria. At each step, the system chooses a navigation feature which maximizes C_{f_i}. We run the experiment using two different weights: $\beta = 0.9$ and $\beta = 0.1$. We want to show the effects of β on conversation efficiency as well as matching quality. In general, a large β guides the navigation towards conversation efficiency and small β guides the navigation towards better matches. We also include the performance of the static navigation approach. We want to show when the combined measure is used, the new approach performs better than the static approach in both conversation efficiency and matching quality. The overall results shown in table 6 indicates that larger β results in better conversation efficiency. When $\beta = 0.9$, in average, it takes 6.05 turns to reach a target, which is less than the 6.84 turns used by the system when $\beta = 0.1$. The average similarity for $\beta = 0.9$ is 8.02, which is less than 8.78, the average similarity score when $\beta = 0.1$ is used. All the differences are statistically signifi-

Table 6: *Results Using Combined Measure*

β	Ave. Exchanges	Ave. Similarity
0.9	6.05	8.02
0.1	6.84	8.78
static	18	na

cant based on the two-sample paired *t test*. In addition, since it takes the static system 18 steps before issuing a data query, both dynamic navigation approaches using the combined measure out-perform the static approach in conversation efficiency. Since almost all the target houses are not in the database, the static approach almost always returns empty set. Thus, both dynamic approaches also outperform the static approach in matching quality.

5 Discussion

In addition to conversation efficiency and user preferences, information navigation can be affected by other factors such as a user's knowledge. In the real estate application, features, such as *price*, are easier for a user to specify than features, such as *school name*, because the likelihood for a user to know his preferred *price range* is much higher than a particular *school name*, especially if he just moved into a new area. Since prompting features that are unknown to a user will result in a few extra turns, it is more efficient if the system prompts for easy features first.

In addition, conversation simulations have its limitations because the target chosen by the system is not a perfect representation of user preferences. For example, in the simulations, we always look for one target house instead of a set of houses in one conversation. This affects the accuracy of our probability estimation. In the future, we want to reproduce these experiments with real human subjects using real house preferences. We want to verify if there is any difference in the results.

6 Conclusion

In the paper, we propose a novel information navigation approach that is sensitive to both search space and user preferences. It out-performs a traditional static navigation approach both in conversation efficiency and matching quality. This approach is particularly suitable for large and complex information systems.

References

D. Albesano, P. Baggia, M. Danieli, R. Gemello, E. Gerbino, and C. Rullent. 1997. Dialogos: A robust system for human-machine spoken dialogue on the telephone. In *Proc. of ICASSP*.

R. Carlson. 1996. The dialog component in the waxholm system. In *Proc. of TWLT*, pages 209–218.

J. Chu-Carroll. 2000. Mimic: An adaptive mixed initiative spoken dialogue system for information queries. In *Proc. of ANLP*.

J. Chu-Carroll D. 1999. Form-based reasoning for mixed-initiative dialogue management in information-query system. In *Proc. Eurospeech*.

D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. 1996. A form-based dialogue manager for spoken language applications. In *Proc. of ICSLP*, volume 2, pages 701–704.

D. Litman and S. Pan. 1999. Empirically evaluating an adaptable spoken dialogue system. In *Proc. of the 7th International Conference on User Modeling*.

K. Papineni, S. Roukos, and T. Ward. 1999. Free flow dialog management using forms. In *Proc. of Eurospeech*, pages 1411–1414.

B. Pellom, W. Ward, J. Hansen, K. Hacioglu, J. Zhang, X. Yu, and S. Pradhan. 2001. University of colorado dialog systems for travel and navigation. In *Proc. of HLT*.

A. Potamianos, E. Ammicht, and J. Kuo. 2000. Dialogue management in the bell labs communicator system. In *Proc. of ICSLP*.

A. Rudnicky, E. Thayer, P. Constantinides, C. Tchou, R. Shern, K. Lenzo, and W. Xu. 1999. Creating natural dialogs in the carnegie mellon communicator system. In *Proc. of Eurospeech*.

M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. Paradise: A framework for evaluating dialogue agents. In *Proc. Of ACL*, pages 271–280.

V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington. 2000. Jupiter: A telephone-based conversational interface for weather information. In *IEEE Trans. Speech and Audio Proc.*, pages 100–112.