# Speech Interface for controlling an Hi-fi Audio System Based on a Bayesian Belief Networks Approach for Dialog Modeling

*Fernando Fernández, Javier Ferreiros, Valentín Sama,*
*Juan Manuel Montero, Rubén San Segundo, Javier Macías-Guarasa, Rafael García*

Speech Technology Group, Dept. of Electronic Engineering, Universidad Politécnica de Madrid
E.T.S.I. Telecomunicación, Ciudad Universitaria s/n, 28040-Madrid, Spain
{efhes, jfl, vsama, juancho, lapiz, macias, rgarcia}@die.upm.es

## Abstract

This paper presents the development of a speech interface for controlling a high fidelity system from natural language sentences. A Bayesian Belief Network approach is proposed for dialog modeling. This solution is applied to infer the user's goals corresponding to the processed utterances. Subsequently, from the inferred goals, missing or spurious concepts are automatically detected. This is used to drive the dialog prompting for missing concepts and clarifying for spurious concepts allowing more flexible and natural dialogs. A dialog strategy which makes use of the dialog history and the system's state is also presented.

## 1. Introduction

We assume dialogue as a communicative process aimed to satisfy some objective. In the context of controlling electronic domestic devices, such goals could be the execution of some commands. In this sense dialogue modeling plays a fundamental role in helping users to reach their dialogue goals efficiently.

Typically, rule based solutions are used. In those solutions the dialog is programmed as a script coded by a set of execution rules. Each of these rules specifies an interaction with the system, [1][2]. Specifically each rule is associated to a specific goal and establishes a set of conditions under which that rule should "fire". When a rule fires, an execution module handles the operation/action. Of course, the rules may trigger the system to ask for more information in order to be able to execute a command or action. However, this is a closed solution since the commands executed are only those ones that exactly match with the conditions of each rule. Therefore this solution allows a limited dialog only in those cases where the user is talking about some goal for which we have designed specific clarification rules.

On the other hand, the Belief Networks (BN) is a stochastic solution that takes several advantages respect to that approach. BN can be applied for mixed-initiative dialog modeling where BN infer the user's informational goals [3]. Subsequently, using the backward inference technique, missing or spurious concepts can be automatically detected based on the inferred goals. This is used to drive the dialog prompting for missing concepts and clarifying for spurious concepts allowing more flexible and natural dialogs. Additionally, the automatic learning of the dialog manager is possible from training data. Moreover, BN allow us to include external knowledge into the BN models from which we manage the dialog. This allows us to take advantage of an expert's knowledge in the application domain.

## 2. System architecture

This is a conversational interface that allows users to drive the Hi-fi system from natural language sentences, differentially from other typical control systems based on simple commands. Thus, users can feel free to give several complex commands from a single sentence. Moreover, they don't have to memorize any command list neither use a closed specific phraseology in order to control the system successfully. The hi-fi audio system we are controlling is a commercial system constituted by a compact disc (with a charger of three discs), two tapes and a radio receiver. This system can be controlled by an infrared (IR) remote control. Instead, users are going to control the Hi-fi system from a microphone. Our interface translates the speech into IR commands in order to carry out some operation or action over the system. This translation is done so that the appropriate IR commands are sent according to the user's intention. Figure 1 shows a block diagram of our interface.

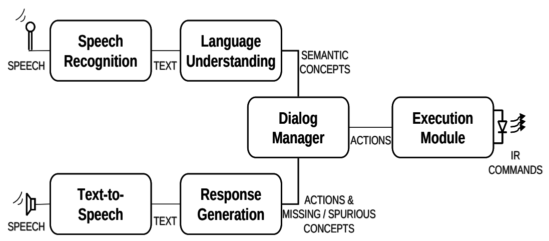

*Figure 1: Block diagram of the interface*

### 2.1. Speech Recognition Module

This module translates the speech into a sequence of words from a continuous speech recognizer based on CD-HMMs.

### 2.2. Language Understanding Module

This module extracts the relevant semantic concepts from the sequence of words output by the recognition module.

First, we have to assign the possible semantic categories to each word present in the sentence according to a semantic dictionary. The dictionary has been defined by an expert in the application domain trying to cover all the semantic categories regarding the control of a Hi-fi system. The semantic categories can be classified into the following groups: actions (to be carried out with the hi-fi system i.e. to play), parameters (that can be configured in the system, i.e. the volume), and values (that can be assigned to the system's parameters i.e. a number).

Last, we apply a set of context dependent rules, designed and handcrafted by the expert, in order to tag each word with the appropriate semantic attributes according to the specific context of the processed sentence. Rules are not eliminatory and their application order goes from the specific to the general. This way we extract the relevant concepts from the sentence as a list of pairs: attribute-value.

### 2.3. Dialog Manager Module

The dialog manager is based on BN. From the list of concepts output by the language understanding module it infers the dialog goals applying Bayesian inference. Then, based on the inferred goals, it applies the backward inference technique to automatically detect the missing and the spurious concepts, prompting the user for the first ones and trying to clarify the latter. This process will be explained in detail later.

From the inferred goals and the confirmed concepts, the dialog manager fills an execution frame for each present action in the user's command. Each action is coded by a frame with three fields: the selected source, the specified parameter and the assigned value.

### 2.4. Execution Module

From the execution frames the execution module determines the set of IR remote commands to be sent to the Hi-fi. This module has to check the system's state before the sending of the IR commands in order to check if it is possible or not to execute the specified action, i.e. it makes no sense to try to switch-off of the system when the system is already off.

Figure 2 represents the system's state.

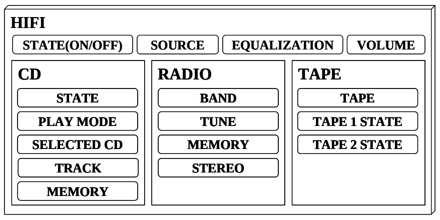

*Figure 2: Hi-fi system's state*

### 2.5. Response Generation Module

This module provides some feedback to the users about the performed actions by the system and the system's interpretation about the user's intention. Moreover, this module defines the appropriate questions from which the system asks the user about some needed information in order to perform a requested action.

As each specific action is associated to a particular dialog goal, this module is constituted by a set of goal specific templates. From these templates the response generation module selects the appropriate text messages to be prompted according to the inferred goal. In order to make the dialogs more natural we select the text message randomly each time.

### 2.6. Text-to-Speech Module

This module synthesizes the speech from the sentence proposed by the generation module in order to give useful feedback to users.

## 3. BN based Dialog Modelling

### 3.1. Overview of the Belief Network Approach

From concepts extracted from semantic parsing, as well as those retrieved from the dialog history, we are going to infer the goal(s) of the user's query. A goal is considered as a specific action over the Hi-fi system, i.e. to set the volume to a specific value. A set of 20 goals and a set of 70 concepts have been defined by an expert in the application domain according to a relevance criterion for the control task.

We are going to develop one BN per goal. A BN is a directed acyclic graph with nodes and arcs where the direction of the arcs represents the probabilistic dependency between two nodes. The arrows of the acyclic graph are drawn from cause to effect. Assuming the basic topology depicted in Figure 3 we are modelling the causal relation between the goal and the concepts.

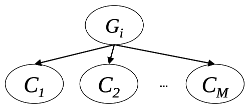

*Figure 3: Basic Topology for a BN*

This topology assumes conditional independence among concepts. Each BN is defined by a specific goal $G_i$ and a set of input concepts $C_j$. We have assumed that the goals and the concepts are all binary, so the concept $C_j$ is true ($C_j$=1) when it is observed in the sentence. In order to avoid too complex models, the expert has selected the concepts with the strongest dependency for each goal as its inputs.

Hence, we have to make N (N=20) binary decisions, with N BN, on the presence or the absence for each goal. From observations extracted from the user's sentence, i.e. $\underline{C}=\{C_1=0, C_2=1,\ldots, C_M=1\}$, we apply Bayesian Inference to obtain the a posteriori probability $P(G_i|\underline{C})$ for each goal (see Equation 1, it simply applies Bayes' Theorem assuming marginal and conditional independence, which is equivalent to a naïve Bayes formulation; M is the number of input evidences). Then we make a binary decision by comparing this probability with a defined threshold.

$$P(G_i=1|\underline{C}) = P(G_i=1)\prod_{k=1}^{M}\frac{P(C_k=c_k|G_i=1)}{P(C_k=c_k)}$$
$$where\ \underline{C}=\{C_1=c_1, C_1=c_2, \ldots, C_M=c_M\}$$

*Equation 1. Bayesian Inference.*

We are going to assume that the goal is present or active if its posteriori probability is greater than the threshold; otherwise the goal is absent. For simplicity, the confidence threshold may be set to 0.5 since (1):

$$P(G_i=1|\underline{C}) + P(G_i=0|\underline{C}) = 1 \quad (1)$$

We are going to assume a multiple goal evaluation scheme, thus multiple goals can be active if they vote positive. Moreover, we can identify the Out Of Domain, OOD, queries. OOD queries are those ones for which all BN vote negative. We have to point out that we have enhanced the basic topology for each goal adding some links between concept nodes, according to the expert's criterion, to model the inter-concept dependencies.

These links introduce certain variation in the probability propagation for goal inference. Details regarding the exact inference algorithm we have used can be checked in [4]. The probabilities involved in the inference process (left hand of Eq.1) have been hand-assigned by the expert too. [5] presents a Minimum Description Length approach for learning topologies automatically. The conditional probabilities for each BN can be estimated simply just tallying the counts from training data.

Next we are going to introduce the "backward inference" technique. This is used to detect automatically missing or spurious concepts based on the inferred goal corresponding to the processed sentence. Consequently, we drive the dialog prompting for missing concepts and clarifying for spurious concepts. From the inferred goal we have to test the network's confidence for each of the input concepts. Now we assume the inferred result, *i.e.* $G_i$=1, as a new evidence that we must add to the observations' vector. Then we apply Bayesian inference again but this time aimed at the estimation of $P(C_i|\underline{C}')$, the updated concept's probability, where $\underline{C}'=\{\boldsymbol{G_i=1}, C_1=0, C_2=1,\ldots, C_M=1\}$.

Based on the value of $P(C_i|\underline{C}')$, we make a binary decision again in order to check whether that concept should be present or absent. Then we compare the result of that decision with the actual occurrence of the concept in the observations' vector. If the binary decision indicates that the concept should be present but it is absent, the concept is labeled as missing and the dialog manager triggers a prompting act. If the binary decision indicates that the concept should be absent but it is actually present in the input sentence, the concept is labeled as spurious and the dialog manager invokes a clarification act.

## 3.2. Dialog Strategy

Figure 4 shows the adopted strategy for dialog modeling. First we build the observations' vector considering the evidences extracted from the semantic concepts. Assuming a starting dialog act we directly apply Bayesian inference. Thus we infer the present dialog goals/actions to be executed according to the user's intention. After evaluating the active goals we proceed to the detection of the missing and spurious concepts. To do this we apply the backward inference based on the inferred goals. In addition, we determine which concepts are confirmed as present by the network.

Subsequently, assuming that the user has provided all the required information regarding the requested actions, we are able to complete for each goal its corresponding execution frame filling it from the confirmed concepts. We store those concepts in the dialog history as consolidated knowledge. Finally, we update the system's state according to the performed actions and inform users about those actions.

### *3.2.1. Partially Observed Goals*

Sometimes people omit certain information which can be perfectly deduced from the dialog context. In such cases the user provides an incomplete observations' vector. As a result several goals can be identified as present in the user's utterance but complete information is not available for all those inferred goals. In order to handle those 'partially observed' goals, we make use of a number of different components: the Dialog History, which maintains a record of the evolving dialogue in the form of a stack of confirmed concepts from backward inference; the Current Dialog Act History, which maintains a similar record to the previous one but it only contains information regarding the current dialog act instead; and the system's state, which contains all the information regarding the system's current setup. Each user's action can be coded by a frame constituted by three elements: a source, a parameter and a value. Therefore we have implemented different strategies in order to recover from each omitted element case.

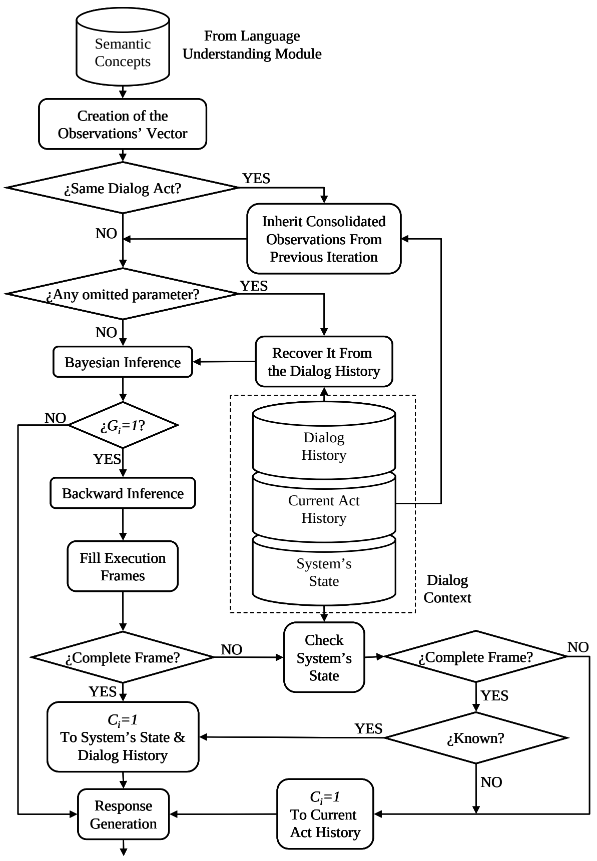

*Figure 4: Dialog Strategy*

Regarding parameters, we assume that a parameter has been omitted when we find a value without an associated parameter. When this occurs ('*YES*' path for '*Any omitted parameter?*' question in Figure 4), we try to match these values to a suitable parameter from the dialog history. There are values and ranges of values that are specific of a particular parameter, i.e. we can't assign a value of 'five' to the 'cd' parameter since the system just has a three discs charger.

Typically, a particular parameter is omitted immediately in those subsequent user's commands which have the aim of assigning a new value to that parameter. Based on this assumption we check the dialog history from more recent to older entries in order to extract, if possible, the closest suitable parameter, i.e. the user instantiates the 'track' parameter for a particular disc from: '*play track six*', subsequently he simply says: '*nine*'. In the last command the parameter has been omitted but it can be perfectly elicited from the dialog history applying the described procedure. If we find any, this is included as an evidence to our observations' vector before applying inference. On the other hand, if we don't find any suitable parameter, we would have to expect the corresponding goal to be active although the parameter is omitted. Otherwise, the user's command would be misunderstood and no action would carry out.

Regarding omitted values no previous search is done in the dialog history before the inference process. Thus, we are referring to those cases where the inference process identifies the present goals properly in spite of the omitted value.

After the completion verification, which obviously results negative, we are going to check the system's state to try to complete that frame. The system's state contains all the information regarding the system's current setup. It also captures the changes derived from the executed actions. Thus, from system's state we are able to complete any dialog frame with a suitable value given a specific parameter.

Regarding how we confirm the user the inherited value, we must state some differentiation depending on the condition of the value and its corresponding parameter. We assume that we can use implicit confirmation just for that updated information induced by the past user's interaction with the system. This information is known by the user since he confirmed it previously, whereas for the rest of information this is not true. In the latter case we must prompt the inherited value using an explicit confirmation procedure. This is clear from the following example: we assume that a user selected the cd two some significant time ago and after that he has performed several actions up to current time but none of them have been either a source or a cd change; if user simply says '*put track number seven from the cd*' there's no doubt that we should inherit the value '*two*' for the '*cd*' parameter from the system's state. The user already knows which cd is selected by the time of processing that command so this information can be elicited directly from the dialog context. On the contrary, if no action for cd selection had occurred previously, the system would ask the user whether the selected cd, recovered from the system's state, is the one that he wants to listen. In this case the user is supposed to be unaware of that information, so the system must explicitly consult him about it. Of course the user is free to select a different value instead of the proposed one.

Last of all, regarding omitted sources we proceed the same way as for omitted values. Thus, we assume that if no explicit change of source has occurred, the omitted source is the last one that was selected and which is stored in the system's state, i.e. if the user says: '*play the tape*', the 'tape' value is omitted but it can be recovered from system's state by checking which tape is selected. Moreover, there are actions which are specific of a particular source, i.e. it is not possible to 'rewind' the radio tune. Thus, in those cases we can recover the source from the active BN.

Finally, after checking the system's state, if we have been able to complete the dialog frame we store in the dialog history both the concepts confirmed from backward inference as well as those inherited ones as consolidated knowledge and start a new dialog act. On the contrary, if the frame is not complete yet, we have to trigger the system to prompt the user for requesting the necessary elements. Consequently, we are going to consider the current dialog act as not finished yet, so we have to store the useful available information before starting a new iteration within the dialog act. Exactly, we are going to store in the current dialog act history only those concepts confirmed from backward inference, neither the spurious nor the optional. Thus, we save the discourse state from the information provided by the user and the system's current intention for confirming or repairing supplied or missing information.

In the next iteration, we are in the same dialog act so those confidence observations from the preceding iteration recorded in the current dialog act history should be inherited directly as part of the evolving observations' vector. From the comparison between the inherited observations and the recently obtained ones the Dialog Manager is able to work out what has been repeated, modified or negated by the user. Therefore the system is able to interpret the user's current utterance in the light of the intention behind its own last utterance and consequently to give an appropriate response.

## 4. Conclusions and Future Work

In this paper we have presented our recent efforts towards the goal of applying BN to spoken dialog systems for mixed-initiative dialog modeling in order to get a more flexible and natural spoken interaction. An approach for a Hi-fi system's controlling domain is described. This can be easily extended to new different domains since portability is ensured following the proposed principles of design.

In the future, we hope to have available training data from we can automatically learn the BN thus avoiding hand-assigned probabilities. We have also incorporated a dialog strategy in order to handle partially observed goals. This strategy is aimed for recovering the omitted information from the dialog context. Suitability has been demonstrated in several cases considering a different omitted element from the dialog frame each time.

We plan to distinguish between long term and short term memory. The aim of this distinction is to give more relevance to the short term observations respect to the long term ones since human mind behaves this way in some sense. Exactly, we would expect the system to automatically reject the erroneous and spurious observations. Thus, we would not need any explicit negation for those observations since they are not referenced by the user anymore.

## 5. Acknowledgements

This work has been partially funded by the Spanish Ministry of Science and Technology under contracts: DPI2001-3652-C02-02 (URBANO-IVANHOE), TIC2003-09192-C11-07 (MIDAS-INAUDITO), and DPI2004-07908-C02-02 (ROBINT).

## 6. References

[1] J.Ferreiros, J.Colás, J.Macías, A.Ruiz and J.M.Pardo, *"Controlling a HIFI with a continuous speech understanding system"*. ICSLP 1998, Sydney, Australia, ISBN 1-876346-17-5

[2] J.Ferreiros, J.Colás, J.Macías, R.Córdoba and J.M.Pardo, *"Control de un equipo de alta fidelidad usando frases habladas de manera natural"*. Iberdiscap 2000, Madrid, Spain.

[3] H.M.Meng, C.Wai and R.Pieraccini. "The Use of Belief Networks for Mixed-Initiative Dialog Modeling", IEE Transactions on Speech and Audio Processing, Vol.11, NO.6, pps. 757-773, 2003.

[4] C.Huang and A.Darwiche, "Inference in belief networks: a procedural guide". International Journal of Approximate Reasoning 1994 11:1-158. USA.

[5] H.M.Meng, W.Lam and K.F.Low. "Learning Belief Networks for Language Understanding". ASRU 1999.