A Stochastic Finite-State Transducer Approach to Spoken Dialog Management

Lluís-F. Hurtado\textsuperscript{1}, Joaquin Planells\textsuperscript{1}, Encarna Segarra\textsuperscript{1}, Emilio Sanchis\textsuperscript{1}, David Griol\textsuperscript{2}

\textsuperscript{1}Department de Sistemes Informàtics i Computació, Universitat Politècnica de València, Spain
\textsuperscript{2}Universidad Carlos III de Madrid, Madrid, Spain
{lhurtado, xplanells, esegarra, esanchis}@dsic.upv.es, dgriol@inf.uc3m.es

Abstract

In this paper, we present an approach to spoken dialog management based on the use of a Stochastic Finite-State Transducer estimated from a dialog corpus. The states of the Stochastic Finite-State Transducer represent the dialog states, the input alphabet includes all the possible user utterances, without considering specific values, and the set of system answers constitutes the output alphabet. Then, a dialog describes a path in the transducer model from the initial state to the final one. An automatic dialog generation technique was used in order to generate the dialog corpus from which the transducer parameters are estimated. Our proposal for dialog management has been evaluated in a sport facilities booking task.

Index Terms: Spoken Dialog Systems, Dialog Management, Stochastic Finite-State Transducers, Automatic Dialog Generation

1. Introduction

The development of spoken dialog systems is one of the main objectives of the spoken language technologies. A dialog system can be seen as a man-machine interface that recognizes and understands the speech input and generates a spoken answer in successive turns in order to reach a goal, such as obtaining an information or carrying out an action. Most of the dialog systems are oriented to restricted domain tasks, mixed initiative, and telephone access.

Different modules usually take part in order to carry out the final goal of a spoken dialog system: speech recognition/understanding module, dialog manager, answer generator, and text-to-speech synthesizer. The use of statistical techniques for the development of the different modules that compose the dialog system has been of growing interest during the last years. This type of methodologies has been traditionally applied within the fields of automatic speech recognition and natural language understanding [1], [2].

Frequently, the dialog managers developed in the spoken language technology area have been manually designed. However, over the last years, the application of statistical methodologies to model the behavior of the dialog manager is providing interesting results in more recent years [3], [4], [5], [6], [7].

Many of these approaches have reasonable behaviors in laboratory environment but present some problems when are applied to more realistic environments. They have to deal with the lack of robustness against unexpected user utterances, or with relevant recognition or understanding errors. Some of these weaknesses are derived from the lack of enough labeled training samples of dialogs for the estimation of the model parameters.

The dialog model proposed in this paper is based on the transduction concept and on the use of Stochastic Finite-State Transducers (SFST) [8]. That is, given a state of the system, and a user turn, a system turn is generated and a transition to a new state is done. Therefore, dialog management is based on the modelization of the sequences of system and user dialog turns pairs, then, a dialog describes a path in the transducer model from its initial state to a final one. The fact that the objective of a dialog manager can be seen as a transduction, and the success of the use of SFST in the machine translation area [8] encouraged us to develop our proposal.

In order to avoid the problem of acquiring and labeling a corpus, we have developed a process in which dialogs are automatically generated. The automatic dialog generation consists of randomly generate sequences of pairs user dialog-act, system dialog-act. We consider a dialog act, not a label that represents the general intention of the turn, but a more detailed information about the turn, that is represented as a register of different levels of information (for example, it can contain information about data that have been supplied by the user in a turn). Once the dialogs are generated a set of correctness criteria is applied in order to discriminate the correct and incorrect dialogs. This generation is similar to [9] also developed in our research group. The automatically generated dialogs are used to learn the SFST parameters.

We have applied these methodologies to the development of a Dialog Manager for a sport facilities booking task. Experiments confirm that this approach have a reasonable behavior and can be used for limited task-domain dialog systems.

Section 2 briefly presents the main characteristics of the EDECAN-SPORT task and the semantic representation chosen for it. Section 3 presents the proposed dialog manager model. Section 4 presents the technique used for the automatic generation of dialogs. Section 5 presents the results of the evaluation of the performance of the proposed dialog manager model. Finally, some conclusions are presented in Section 6.

2. The EDECAN-SPORT task

Within the framework of the EDECAN project [10], a task called EDECAN-SPORT consisting of mixed-initiative dialogs for booking sport facilities using spontaneous speech has been designed. For the acquisition of a dialog corpus for this task we followed the process described below.

We analyzed human-human dialogs provided by the sports area of our university, which have the same domain defined for the EDECAN-SPORT task. From these dialogs we defined the semantics of the task in terms of dialog acts for both the user utterances and system answers, and we labeled these dialogs.

The definition of the semantics of the EDECAN-SPORT task has been carried out considering the different functionalities required for the booking system and the information required to complete them. For the user turns (the set of user dialog acts) we defined four task-dependent concepts.
(DC): (Availability, Booking, Booked, Cancellation), three task-independent concepts (IC) (Affirmation, Negation, and Not-Understood) and six attributes (AT) (Sport, Hour, Date, Court-Type, Court-Number, and Order-Number). Concepts represent the actions the user can ask for, and attributes provide additional information to the system and can be seen as restrictions imposed by the user to achieve the dialog goal. An example of the semantic interpretation of an input sentence is shown below:

_I want to book a paddle court for tomorrow._

**Semantic Representation:**

**Booking**

-Sport: paddle
-Date: tomorrow

The labeling of the system turns in terms of system dialog acts, is similar to the labeling defined for the user turns. A total of 21 concepts has been defined: Task-independent concepts (Opening and Closing); concepts used to inform the user about the result of a specific query (Availability, Booking, Booked, and Cancellation), concepts defined to require the user the attributes that are necessary for a specific query (Sport, Date, Hour, Court-Number, and Court-Type), concepts used for the confirmation of concepts (Confirmation-Availability, Confirmation-Booking, Confirmation-Booked, Confirmation-Cancellation), and attributes (Confirmation-Sport, Confirmation-Date, Confirmation-Hour, Confirmation-CourtType).

We defined two more concepts to ask the user to choose between several alternatives when trying to make a booking or a cancellation (Booking-Choice, and Cancellation-Choice).

A total of six attributes has been defined (Sport, Court-Type, Court-Number, Date, Hour, and Availability-Number). These attributes are used to provide additional information to the user.

An example of the labeling of a system turn is shown below:

_To play paddle tomorrow, there are two courts: court number 3 at 10:00 and court number 1 at 16:00. Please choose one._

**Semantic Representation:**

**Booking-Choice**

-Sport: paddle
-Date: tomorrow
-Hour: 10:00 16:00
-Court-Number: 3 1

During the corpus acquisition process, we used a specific Wizard of Oz (WOz) to play the role of the natural language understanding module and a second WOz to control the dialog manager. Using these two WOz allows us to obtain after the acquisition process not only the dialog corpus, but also the dialog acts corresponding to the labeling of the user and system turns.

A set of 143 dialogs, by 16 different speakers from different origins, was acquired for this task. The languages involved in the acquisition were Spanish, Catalan and Basque. A set of 15 types of scenarios was defined in order to cover all the possible use cases of the task.

The information available for each dialog consists of four audio channels, the transcription of the user utterances (with an average of 4.9 user turns per dialog and 6.7 words per user turn) and the semantic labeling of the user and system turns.

2.1. The Application Manager

In some dialog systems, the dialog manager takes its decisions based only on the information provided by the user in the previous turns and its own model. The main difference between some of these slot-filling tasks and the task defined for the EDECAN-SPORT project is that in this last task the dialog manager not only provides information but also modifies the application data (i.e. after booking or canceling a court).

Therefore, the dialog manager generates the following system answer taking into account not only the information provided by the user, but also the information generated by the module that controls the sport facilities booking application (that we call Application Manager, AM).

The AM performs the queries to the information system and updates it when it is necessary. For example, the information system for the EDECAN-SPORT task must be updated when Booking or Cancellation actions have been performed. Then, the result of the queries to the AM has to be considered to generate the system answer. For instance, in order to book the facilities (i.e. a tennis court), if there is not any available court, the system can suggest a change in the user restrictions (i.e the AM verifies if it is possible to perform the booking changing the hour). In case of only one court available, the system confirms if everything is correct before making the booking. Finally, if there is more than one available court, the system asks which court has to be booked as shown in the previous example.

3. The dialog manager model

In a dialog system, the Dialog Manager (DM) is the module devoted to choose the best system answer according to its dialog model during the dialog sequence. We consider a dialog as a sequence of pairs (ui, ai), i = 1, n, where ui is the user utterance at time i and ai is the answer of the system to this utterance. The system answer ai is selected taking into account not only ui but also all the information provided by the user throughout all the dialog sequence. All this information (in terms of concepts and attributes) is stored in a data structure we call Dialog Register (DR). DRi is a summary of the information provided by the user until time i.

In order to reduce the DR variability, we assume that the exact value of an attribute is not necessary for the DM to choose the next system answer. These values are used in the AM query or in the construction of the output sentence of the system. However, the only information necessary to determine the next system answer is the presence or absence of concepts and attributes and their confidence scores. Therefore, the information we used from the DR is a codification of each of their fields in terms of three values, (not-supplied, low-confidence, high-confidence).

In order to summarize all the dialog history it is not enough to use the Dialog Register. Therefore, we define and additional data structure in such a way that contains all the information required by the dialog manager to select the next system answer. We call this structure Dialog State (DS). A DS includes: i) the Dialog Register, ii) the last request to the application manager, iii) the cardinality of the last application manager result (with four possible values: no-row, one-row, two-rows, three or more rows), and iv) the last system answer.

We have developed a statistical DM based on the use of a Stochastic Finite-State Transducer (SFST) [8]. A SFST is defined formally by a 6-tuple (Q, Σ, Δ, q0, p, f), were Q is a set of states, Σ is the input alphabet, Δ is the output alphabet, q0 is the initial state, the function p : Q × Σ × Δ × Q → [0, 1] is the transition probability distribution, and the function f : Q → [0, 1] is the final-state probability distribution. In our approach:

- Q contains all the possible dialog states represented as stated above (DS).
• The input alphabet $\Sigma$ represents all the allowed user utterances. Given a task semantic definition, we define $\Sigma$ as $\{IC \times DC \times 2^{AT}\}$. That is, all the possible combination of one task-independent concept (IC), one task-dependent concept (DC), and any quantity of attributes (AT).

• The output alphabet $\Delta$ includes all the system answers defined for the task, that is, the set of system dialog acts.

• There is only one initial state $q_0$. All dialogs start at the same point.

• $p(q, u, a, q') = Pr(u, a, q'|q)$ is the probability of transition from $q$ to $q'$ by observing $u \in \Sigma$ and emitting $a \in \Delta$. In our approach this probability is independent of the destination state, then, $p(q, u, a, q') = p(q, u, a) = Pr(u, a|q)$. The new state is calculated by an update function consisting of, basically, adding the information provided by the user in $u$ to the DR and the cardinality of the result of the query to the Application Manager.

• Once the dialog reaches a final state the dialog ends. Therefore, $f(q) = 1$, for all states that can be reached with a system answer that involves the end of the dialog and $f(q) = 0$ for all the others.

Due to the determinism of the model, $p$ can be estimated from a labeled dialog corpus as:

$$p(q, u, a, q') = Pr(u, a, q'|q) = \frac{C(q, u, a)}{C(q, u)}$$

where $C(q, u, a)$ is the number of times that being at dialog state $q$ and observing the user utterance $u$ the system answer was $a$; and $C(q, u)$ is the number of times that being at dialog state $q$ the user utterance $u$ was observed.

In our proposed approach, the selection of the best next system answer at time $i$ ($a_i$) is made by means of the following local maximization:

$$a_i = \arg \max_{a_i \in \Delta} p(q_{i-1}, u_i, a_i, q_i) = \arg \max_{a_i \in \Delta} Pr(u_i, a_i|q_{i-1})$$

The dialog ends when a final state $q_f$ is reached. From that point of view, a dialog can be seen as a path in the transducer from the initial state $q_0$ to the final state $q_f$.

4. Automatic dialog generation

In order to learn the SFST parameters, the acquisition of a corpus of labeled dialogs is required. Due to the high cost of acquiring dialogs with real users, we propose an approach for the automatic generation of dialogs that only requires the semantic definition of the task and a set of criteria to determine the correctness of the generated dialog.

Our approach is based on the simulation of the main components of a spoken human-machine dialog system. It uses a user simulator, a communication channel simulator, an application manager simulator, and a dialog manager simulator.

The user simulator randomly chooses its goal at the beginning of the dialog. One concept is selected from the task-dependent concept set (Availability, Booking, Booked, or Cancellation). Then, a subset of the attributes is added to the goal. It is not necessary to assign specific values to the attributes because no real access to the information system will be done during the dialog generation.

While the dialog is being generated, each user turn is randomly selected from the allowed user utterances set $\Sigma$. A confidence value in $[0,6,1]$ is attached to each concept, attribute, and attribute value in the user turn. The attribute values are all set to Correct.

The communication channel simulator is used in order to modify the user turn. A random number in $[0,1]$ is generated for every concept, attribute, and attribute value in the turn. If the generated value is greater than the confidence attached to that field we change the value to Error; however, the confidence values from the turn remain unchanged. With this approach, the fields with lower confidence are more likely to be confused than the ones with higher confidence. During the dialog generation, the communication channel simulator introduces $20\%$ of errors in average. The confidence interval was selected manually to be similar to the error rate obtained with our Spoken Language Understanding module.

Like the user simulator, the dialog manager simulator chooses at each turn a random answer. If the selected answer involves an interaction with the application manager, the application manager simulator is launched. This module randomly chooses the cardinality of the query to the information system.

When the dialog ends it is time to decide if it is correct or not. We have defined a set of simple criteria in order to automatically determine the correctness of a dialog. A dialog is considered to be unsuccessful if one of the following conditions takes place:

• The dialog exceeds the maximum number of turns. The maximum number of turns allowed for the simulated dialogs is defined to be a slightly higher than the average number of turns of the dialogs acquired with real users.

• The dialog manager simulator has modified the information system with information marked as Error or with attributes not present in the user goal.

• The dialog manager simulator has chosen an action that needs information not provided by the user. For example: asking for a confirmation of an attribute that the user has not said.

• The dialog manager simulator performs an update of the application information system not related to the user goal.

Otherwise, the dialog is considered correct and is added to the dialog corpus.

Using this methodology, a corpus of 200,000 successful dialogs has been obtained from a set of more than 9 billions of automatically generated dialogs.

5. Evaluation

A SFST dialog manager was learned using the corpus acquired with the automatic dialog generator technique described in Section 4. The EDECAN-SPORT corpus described in Section 2 was used as test set to evaluate the behavior of this dialog manager with real users.

In order to evaluate the performances of the Dialog Manager we have defined the following measures:

• $\#D$: Correct Dialogs. Number of dialogs successfully generated. These are the dialogs that meet the correctness criteria and are used to learn the SFST.

• $|Q|$: Number of states of the transducer.
• **Out**: Out of model turns. Percentage of dialog turns in the test set for which the DM has no answer.
• **ETR**: Exact turn rate. Percentage of turns in the test set for which the answer of the transducer is equal to the one produced by the WOz during the real acquisition.
• **EDR**: Exact dialog rate. Percentage of dialogs in the test set that in all their states the answer of the transducer is equal to the one produced by the WOz during the real acquisition.
• **ADR**: Accepted dialog rate. Percentage of dialogs in the test set accepted by the transducer. That is, the percentage of dialogs in the test set for which a path between the initial state and their final states in the transducer exists.

Table 1 shows the evolution of the result of the Dialog Manager evaluation with regard to the incorporation of new dialogs.

<table>
<thead>
<tr>
<th>#D</th>
<th></th>
<th>Out</th>
<th>ETR</th>
<th>EDR</th>
<th>ADR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>3,121</td>
<td>0.456</td>
<td>0.343</td>
<td>0.007</td>
<td>0.084</td>
</tr>
<tr>
<td>5,000</td>
<td>9,543</td>
<td>0.241</td>
<td>0.573</td>
<td>0.105</td>
<td>0.259</td>
</tr>
<tr>
<td>10,000</td>
<td>14,375</td>
<td>0.176</td>
<td>0.640</td>
<td>0.168</td>
<td>0.399</td>
</tr>
<tr>
<td>50,000</td>
<td>32,332</td>
<td>0.096</td>
<td>0.701</td>
<td>0.245</td>
<td>0.601</td>
</tr>
<tr>
<td>100,000</td>
<td>43,163</td>
<td>0.069</td>
<td>0.753</td>
<td>0.266</td>
<td>0.727</td>
</tr>
<tr>
<td>150,000</td>
<td>50,211</td>
<td>0.069</td>
<td>0.749</td>
<td>0.266</td>
<td>0.727</td>
</tr>
<tr>
<td>200,000</td>
<td>55,645</td>
<td>0.067</td>
<td>0.750</td>
<td>0.266</td>
<td>0.734</td>
</tr>
</tbody>
</table>

Table 1: Evolution of the result of the Dialog Manager evaluation with regard to the incorporation of new dialogs.

Table 1 shows a low accepted dialog rate, even when all the training set is used (0.734). In our approach, when during a dialog a state \( q \) is reached and \( C(q, u) = 0 \) where \( u \) is the next user turn, the DM is unable to continue. In order to deal with this problem a kind of smoothing technique has been defined. The DM look for the first state \( q' \) in the path described by the dialog for which \( C(q', u) > 0 \).

Table 2 shows the evolution of the result of the Dialog Manager evaluation applying the smoothing technique with regard to the incorporation of new dialogs.

<table>
<thead>
<tr>
<th>#D</th>
<th></th>
<th>Out</th>
<th>ETR</th>
<th>EDR</th>
<th>ADR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>3,121</td>
<td>0.276</td>
<td>0.423</td>
<td>0.014</td>
<td>0.196</td>
</tr>
<tr>
<td>5,000</td>
<td>9,543</td>
<td>0.159</td>
<td>0.647</td>
<td>0.182</td>
<td>0.441</td>
</tr>
<tr>
<td>10,000</td>
<td>14,375</td>
<td>0.053</td>
<td>0.746</td>
<td>0.343</td>
<td>0.783</td>
</tr>
<tr>
<td>50,000</td>
<td>32,332</td>
<td>0.023</td>
<td>0.769</td>
<td>0.371</td>
<td>0.895</td>
</tr>
<tr>
<td>100,000</td>
<td>43,163</td>
<td>0.000</td>
<td>0.817</td>
<td>0.413</td>
<td>1.000</td>
</tr>
<tr>
<td>150,000</td>
<td>50,211</td>
<td>0.000</td>
<td>0.813</td>
<td>0.413</td>
<td>1.000</td>
</tr>
<tr>
<td>200,000</td>
<td>55,645</td>
<td>0.000</td>
<td>0.813</td>
<td>0.413</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 2: Evolution of the result of the Dialog Manager evaluation applying the smoothing technique with regard to the incorporation of new dialogs.

Both tables show that all measures improve when the size of the training set increases until 100,000 dialogs; from this size the results do not change. This could be because the strategy to obtain the corpus automatically generated from which the model is estimated, has got a variability in dialogs that can not be improved by adding more than 100,000 dialogs. Applying the smoothing technique the model gets a complete coverage of the test set, and in the 81.3% of turns the answer of the DM is exactly the same as the reference (the answer of the WOz during the acquisition process). The difference between ADR and EDR values could be due to the fact that there are more than one dialog to achieve the same user goal.

6. Conclusion

In this paper, we have presented an approach for the development of stochastic Dialog Managers learned from training samples. The proposed approach is based on the use of a Stochastic Finite-State Transducer. An automatic dialog generation technique has been used to generate the corpus from which the SFST parameters have been estimated. Some experiments, using a sport facilities booking task, have been performed to test the behavior of the approach. The results show the satisfactory operation of the approach.

We are interested in the study of the behavior of our Dialog Manager with real users, and in the application of active learning techniques to detect and to correct wrong answers of the system (unseen situations) during the real dialogs. An improvement in the smoothing technique will be needed in order to solve this lack of coverage of the model and to correct wrong answers. Finally, we want to study the adaptation of the proposed approach to other spoken dialog tasks.

7. Acknowledgements

Work partially supported by the Spanish MEC and FEDER under contract TIN2008-06856-C05-02

8. References