Dialog Studio: An Example Based Spoken Dialog System Development Workbench

Sangkeun Jung, Cheongjae Lee, Gary Geunbae Lee

Department of Computer Science and Engineering
Pohang University of Science and Technology
{hugman,lcj80,gblee}@postech.ac.kr

Abstract

In recent years, spoken dialog systems have been exploited by many researchers, but it is still too early to apply it for practical use in commercial fields. In addition to the inaccuracy of each component of dialog systems including automatic speech recognizer (ASR), spoken language understanding (SLU), and dialog management (DM), the inconvenience and the high cost of maintaining and upgrading the spoken dialog systems are one of the biggest problems that prevents the use of spoken dialog systems in commercial applications. This paper introduces a dialog workbench tool, *Dialog Studio,* which is a back-end workbench system for example-based spoken dialog systems. It is designed for reducing human efforts of tutoring the dialog systems with new dialog examples, supporting semi-automatic learning by recommending a dialog administrator with proper candidate answers and synchronizing all dialog system's components for upgrading spoken dialog systems.

1. Introduction

One of the biggest restrictions that we face when we are trying to use a spoken dialog system in a practical field is the difficulty of maintaining and upgrading the spoken dialog system. There were several components such as ASR, SLU and DM in the system, and a lot of engineering works are needed for modifying and upgrading each models. Even for the system developers, the work is very laborious and tricky, thus much harder for non-experts.

To reduce the laborious work, several researches have been conducted. For the rapid application development, [1] introduced CSLU Toolkit and [2] developed SGStudio to manage the schemes in the dialog. Moreover, [3] developed SUEDE tool to help non-experts in developing a user interface in a spoken dialog system.

These tools provided help in reducing human efforts, but these works were focused on developing the dialog system easily, not on maintaining or upgrading the dialog system robustly, and the target is also limited in one of the components of the spoken dialog system.

To reduce the engineering work and to upgrade the whole components including ASR, SLU and DM together, we developed a back-end spoken dialog system workbench tool, ***Dialog Studio***.

We will introduce our example-based spoken dialog system [4] that the Dialog Studio supported in section 2. Detail workbench strategies are proposed in section 3. Implementation information will be shown in section 4, and conclusions will be drawn in section 5.

2. Example Based Spoken Dialog System Overview

Our speech recognizer was developed based on open source HTK (Hidden Markov Model Toolkit) [5]. The ASR (Automatic Speech Recognizer) generates the n-best recognition list for our SLU (Spoken Language Understanding) module.

The SLU module was constructed by a concept spotting approach [6] which aims to extract only the essential factors for predefined meaning representation slots (e.g. channel, program, and genre for an EPG guide domain). Although this method was regarded as a partial understanding approach, we can acquire enough information to lead the dialogue from the extracted semantic slot values because the slots are properly designed for a specific domain-oriented understanding task.

The dialogue manager module achieves a task completion goal of a specific domain task through a series of interactions with users by using the results of the SLU dialog frames. The dialogue manager triggers the system concepts to generate the system responses using a discourse manager and domain expert modules based on a current situation of the dialogue.

After determining the correct system concepts, the language generation module makes generic system responses including slot names. Then, the system utterances are generated by filling slots in a domain-specific generation module linked to the domain expert. The dialog manager employs three different classes of the rules:

- Situation-action rules: rules for describing the system's actions under the current situation. The current situation designates user intention, semantic frame and discourse history.
- Constraint-relax rules: rules for relaxing the constraints on database queries to broaden the search scope.
- Frame-reset rules: rules for restarting a new dialogue frame for the case of domain switching and dialogue closing.

As shown in Fig. 1, using these rules, the discourse manager decides only generic dialogue strategy which is domain-independent. This generic strategy is inherited to the domain expert that has its own dialogue history, semantic frame, and situation-based rules to manage the domain specific dialogs.

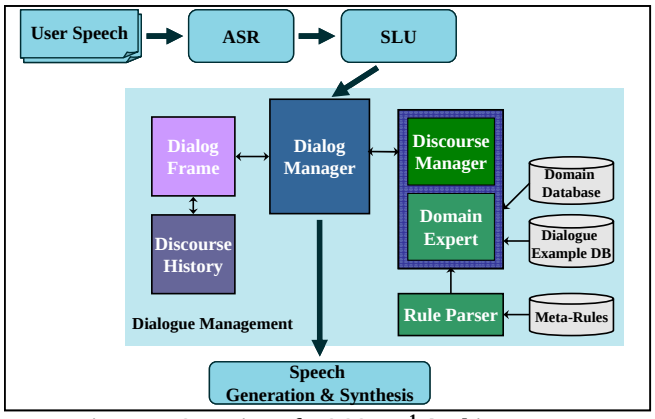

Figure 1: Overview of POSSDM[1] Architecture

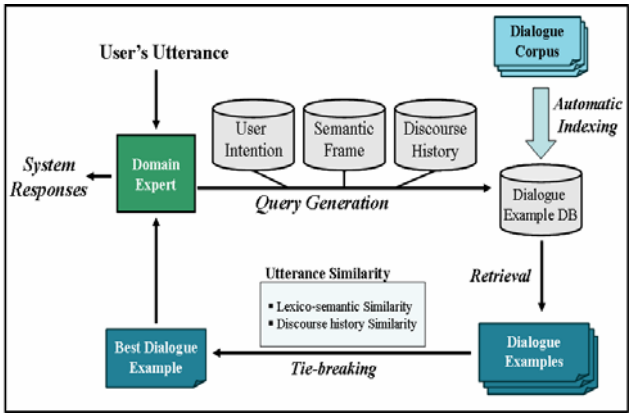

Figure 2: A strategy of an example-based dialog modeling

For more efficient and domain portable dialog model construction, we devised an example-based technique which automatically generates system responses from an example dialogue corpus [4].

For a dialog model, we should make the query key to search the matched dialog examples. The constraints of the search consist of the current dialog situation such as user intention, semantic frame and discourse history. However, in some cases, we need to relax the constraints to do a partial match by using only the user intention from the SLU module.

When the retrieved examples are not unique, we choose the best one using the utterance similarity. The utterance similarity values include the lexico-semantic similarity and the discourse history similarity. The lexico-semantic similarity is defined as an edit distance between the utterances of the users and the matched examples. The degree of the discourse history similarity is a cosine measure between the binary vectors which represent a similarity of the two previous dialog histories. Fig 2 illustrates an overall strategy of the example-based dialogue modeling.

Although the dialogue examples can generate appropriate responses for most of the dialogue situations, some situations require the meta-rules to lead the dialogue. For example, if the retrieved dialogue example result is absent, the system should give an alternative suggestion. To deal with these special situations, some manually designed situation-based meta-rules were used together.

3. Dialog Studio: Workbench for Robust Spoken Dialog System Maintenance.

The critical problem of dialog systems in commercial fields does not lie in the performance of the dialog system but in the feasibility of maintaining the system in a fast and easy way. Since a spoken dialog system is a total integrated ensemble of each separate but inter-related module including ASR, SLU, and DM, we should re-model each module for maintaining the performance of a dialog system. And this re-modeling work is extremely time consuming and requires tremendous human efforts.

Therefore, to use the spoken dialog systems commercially, we need a well designed method for maintaining the dialog system which allows the dialog system administrator to re-model the whole dialog system components. Namely, we need a workbench system which is specially designed for maintaining, tutoring and synchronizing all dialog components in an easy way. For this, we developed a ***Dialog Studio*** which is the name of the back-end workbench system for our example-based spoken dialog system.

We have considered the advantages of the spoken dialog system workbench from the perspective of its practicability. These advantages can be summarized in the following four properties:

- System tutoring of the new dialog situations
- Upgrading all models (ASR, SLU and DM) with least user efforts
- Synchronizing all dialog components
- Semi-automatic learning of the dialog models

3.1. Dialog Workbenching Flow

As shown in Fig 3, our workbench system starts from editing the dialog examples. The dialog system administrator is only required to add new dialog examples with system recommendation or verify auto-generated dialog example candidates that the workbench system generated. Building up new ASR's pronunciation dictionary, Language Model (LM), new SLU model, and new example-based dialog model are all automatically executed under the control of the workbench system.

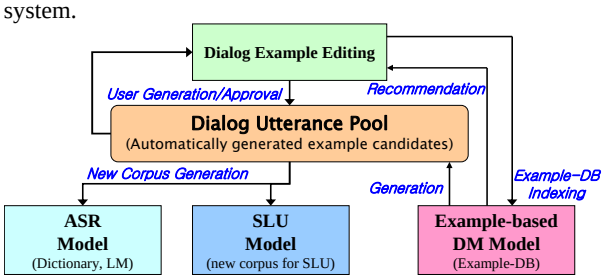

Figure 3: Whole flow of the dialog workbenching.

[1] Postech Situation-based Dialog Manager

3.2. Dialog Example Editing

Dialog example editing is the way of the system tutoring. In our system, if we add new examples which describe the situation including user and system utterance, SLU information, and dialog history information, the workbench system automatically builds up new example-based DM model by example-DB indexing. Adding or editing dialog examples causes the dialog system to handle new types of dialog patterns in an easy way. Table 1 shows how new dialog examples can be managed by adding the proper example using the Dialog Studio.

<table>
<tr><td colspan="4">오늘10 시에는 뭐 하니?
(What's on TV at 10 pm today?)</td></tr>
<tr><td rowspan="6">Example Editing</td><td>Dialog_act</td><td colspan="2">Wh_question</td></tr>
<tr><td>Main_goal</td><td colspan="2">Search_program</td></tr>
<tr><td>Date</td><td colspan="2">Today</td></tr>
<tr><td>Time</td><td colspan="2">10 pm</td></tr>
<tr><td>Filled_slot</td><td colspan="2">{"date","time"}</td></tr>
<tr><td>System_action</td><td colspan="2">Inform(Channel, Program)</td></tr>
<tr><td colspan="4">After re-indexing Example-DB with the added example, if user re-asks
"오늘9 시에는 뭐 하지?"
(What's on TV at 9 pm today?)
[note the Korean lexical form is different in verb-ending]</td></tr>
<tr><td>SQL</td><td colspan="3">SELECT * FROM TVGUIDE_DEDB WHERE dialog_act = "wh-question" AND main_action = "search_program" AND date="1" AND time = "21" AND slot_vector = "[0,0,0,1,1,0,0,0];</td></tr>
<tr><td rowspan="3">Example Candidates</td><td colspan="2">Candidate Utterances</td><td>L</td><td>H</td><td>U</td></tr>
<tr><td colspan="2">[date] [time]에는 뭐 해?</td><td>0.58</td><td>1.0</td><td>1.58</td></tr>
<tr><td colspan="2">[date] [time]에는 무슨 프로해?</td><td>0.16</td><td>1.0</td><td>1.16</td></tr>
</table>

Table 1: After indexing new dialog example in Example DB, similar user utterances can be managed properly. DM chooses most similar example by estimating the utterance similarity (U) which is a sum of the lexico semantic similarity (L) + discourse history similiary (H).

The system recommendation assists in adding a new dialog example. After adding the new user's utterance, the workbench system analyzes the utterance with the current SLU and DM model and generates the n-best system responses. The administrator can add a new system's response by approving or disapproving.

In addition to adding the dialog example manually, Dialog Studio can provide automatically generated user-system utterance pairs (DUP; Dialog Utterance Pool) using the current DM model. From these semi-automatically tagged dialog examples, the administrator can easily add new dialog examples by just approving or disapproving

3.3. Dialog Utterance Pool

Dialog Utterance Pool (DUP) is a set of automatically generated dialog candidates. DUP is generated by using the current dialog examples stored in the current Example DB as well as newly added dialog examples. DUP is an intermediate result for building a new pronunciation dictionary and a language model for ASR, and a new SLU model.

Below listed are the reasons for generating candidates of possible user's utterances and corresponding system responses automatically:

- Manually added examples are never enough to cover the practical user's utterances.
- Dialog system models including ASR, SLU and DM are easily over-fitted if we only use the manually tagged examples.
- Automatically generated examples and the annotation-recommendation support semi-automatic learning of the models.
- Synchronization in each module can be easily implemented by using the DUP.

Synchronization of each module is one of the most important features of a robust spoken dialog system. The ASR and SLU should be well-fitted to what DM can handle. We do not need extra vocabularies or utterance patterns that DM cannot manage. In other words, the ASR, SLU and DM should be modeled to handle the same patterns of the language. This synchronization helps the total upgrade of the spoken dialog system

Automatic dialog utterance generating algorithm and the examples are shown in Table 2 and Table 3 respectively..

Possible Utterance Structure Generation Step:

1. From the manually tagged user utterances, build up 2-gram, 3-gram structure models using a class-based LM.
2. Using the n-gram structure models, generate the n-best possible structure list.

Filling the Generated Structure Step:

1. For each utterance structure, fill the structure with the corresponding vocabularies using the dictionary.
2. Add functional words around the structures.

Removing the infeasible utterance Step:

Using the domain knowledge DB, cut off the utterances which are impossible to occur.

Table 2: Algorithm for generating the candidate dialog utterance examples

As shown in Table 3, the automatically generated examples have not only utterances but also tagged information. It is possible that the annotation might be wrong. Thus, these candidate utterances and the corresponding system responses which are recommended by the current DM should be finally approved by the dialog system administrator.

3.4. ASR Maintenance

Usually, we need an acoustic model, a language model and a pronunciation dictionary to build a new domain speech recognizer. However, in this research we have not considered building an acoustic model in our workbench system since the acoustic model can be reusable after being generally trained. Therefore, we have only considered building a new pronunciation dictionary and a domain specific LM.

Both the dictionary and LM are automatically generated using the DUP. Grapheme to phoneme (G2P) conversion generates the pronunciation for each word unit of the utterances among the DUP [7]. The 2-gram and 3-gram LM's are also generated from the user's utterances among the DUP.

N^{th} utter : "I want to watch drama Hae-Sin around 9 pm"	
... [genre = drama] , [program_name = Hae-Sin] , [time = 9pm] ...	
Structure Generation Step	3-gram of the structure : genre→program_name→ time program_name → genre → time Time→Channel→ program ...
Filling Step	"**Drama**, I want to watch **X-file** at **11 pm**" "**X-file**, **Movie** around **9 pm**" "Around **9 pm**, On **KBS**, any **news**?" ...
Removing Step	Domain Knowledge : [X-file, drama, KBS,11pm] [news_9, news, KBS, 9pm] "**Drama**, I want to watch **X-fill** at **9 pm**" **[genre=drama]→[prog_name=X-file]→[time=9pm]** "Around 9 pm, On KBS, any news?" **[time=9pm] → [channel=KBS]→[genre=news]**

Table 3: Examples of the automatically generated candidate dialog utterances in a TV-guide domain

3.5. SLU Model Upgrade

Like ASR maintenance, SLU model upgrading is executed using the DUP. However SLU maintenance needs correctly annotated corpus while DUP's automatically generated dialog utterances may contain incorrectly tagged utterances. To obtain robust SLU model, we used the combination of active learning and semi-supervised learning method [8] for semi-automatic tagged corpus in DUP.

Using the current SLU model, we predict the semantic structure of the utterances of the DUP, and estimate the prediction confidence. If the confidence is over the threshold, the utterance and predicted labels are added to an augmented training corpus. However, if the confidence is lower than the threshold, the utterance and labels are sent to the administrator to be checked.

The difference between our combining method and [8] is that, instead of using the raw data, we used the semi-automatically tagged data in DUP. Therefore we can use this semi-automatic tagged answer as another indicator for estimating the confidence score. We voted an answer out of comparing the SLU prediction with the semi-automatic tagged result in DUP. The voting weight should be decided carefully according to the size of the corpus which is used for current SLU model training.

4. Implementation

We implemented the Dialog Studio for Electronic Program Guide (EPG) domain for Digial TV. Our example-based dialog system [4] was developed using standard C++ library and can run under both Linux and Windows platform.

Dialog Studio was also developed using standard C++ library for the engine part, but the GUI part is coded using Microsoft Visual Studio's MFC library. The screen shot in figure 4 shows the dialog example editing screen.

5. Conclusion

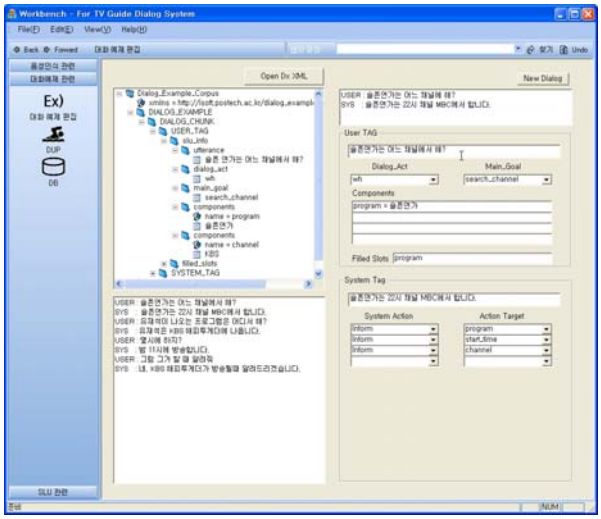

Figure 4: Screen shot of the dialog example editing in Dialog Studio.

We examined the qualities of the spoken dialog system workbench such as the system tutoring, synchronizing all dialog components, reducing human efforts and semi-automatic learning ability. We implemented Dialog Studio which has a new workbench strategy that supports the above four qualities. System tutoring is implemented by dialog example editing with system's recommendation. By devising the intermediate dialog utterance pool, we not only implement the semi-automatic learning but also synchronize all dialog components successfully. The greatest advantage of our Dialog Studio is that it requires very little human effort to maintain and upgrade the whole spoken dialog system.

6. Acknowledgements

This work was supported in part by MIC & IITA through IT Leading R&D Support Project.

7. References

[1] CSLU Toolkit, http://cslu.cse.ogi.edu/toolkit/

[2] Wang, Y and Alex Acero. SGStudio: Rapid Semantic Grammar Development for Spoken Language Understanding. *Proceedings of the Eurospeech Conference. Lisbon, Portugal. September, 2005.*

[3] Anoop, K. S., Scott R. K., Chen., J., Landay, J. A., Chen, C., "SUEDE: Iterative, Informal Prototyping for Speech Interfaces." Video poster in *Extended Abstracts of Human Factors in Computing Systems: CHI 2001*, Seattle, WA, March 31-April 5, 2001, pp. 203-204.

[4] Lee, C., Jung, S., Eun, J., Jeong, M. and Lee, G. G., A Situation-based Dialog Management Using Dialog Examples. *To appear in the International Conference on Acoustics, Speech, and Signal Processing, 2006.*

[5] Hidden Markov Toolkit (HTK), http://htk.eng.cam.ac.uk/

[6] Eun, J., Jeong, M. and Lee, G. G, A multiple classifier-based concept-spotting approach for robust spoken language understanding. *Proceedings of European Conference on Speech Communication and Technology 2005*, pages 3441-3444.

[7] Kim, B., Lee, G.G, Lee, J., "Morpheme-Based Grapheme to Phoneme Conversion Using Phonetic Patterns and Morphophonemic Connectivity Information," *ACM Transactions on Asian Language Information Processing, Vol. 1, No. 1: 65-82, 2002.*

[8]Tur, Gokhan, Tur, Dilek. H, Schapire, R. E., Combining Active and Semi-Supervised Learning for Spoken Language Understanding. *The Journal of Speech Communication, Vol. 45, No. 2, pp. 171-186, 2005.*