Dialog Management for Conversational Applications

Vidhya Ramanujam, Greg Purdy, Carl Osipov, Rajesh Balchandran, Linda Boyer
IBM TJ Watson Research Center, Route 134, Yorktown Heights, NY 10598
vidhyar, gps, osipov, rajeshb, lboyer@us.ibm.com

Abstract

In this paper, we describe the features of a dialog management system for conversational mixed-initiative natural language applications. This system, called the Conversational Interaction Manager (CIM) was designed to meet the complex needs of conversational systems that promote best practices in speech user interface design while facilitating rapid application development through re-usable features and rich default behavior.

1 Introduction

Speech based mixed-initiative Natural Language Understanding (NLU) dialog processing consists of three tasks – recognizing *natural language* requests, interpreting them and managing the dialog flow[1]. The dialog flow is controlled by a Dialog Management system, which uses the current user response along with application information to determine the next step in the dialog. In a mixed initiative system, the dialog flow often varies from the pre-defined flow based on the current user request and previous history of requests. The dialog management system has to allow for task switching and also provide the ability to inherit user input and information provided during one task and uses it in another related task.

2 Dialog Manager Overview

This dialog manager also called as Conversational Interaction Manager (CIM) is forms based and has an event-driven programming model. This uses the concepts of fields and forms to handle speech based mixed-initiative dialog. The CIM can also be used to perform dialog management functions in a text-based (non-speech) environment such as a web chat. An application developed within the CIM framework has three distinct sections described below.

Form markup contains the definitions of the fields, goals, and events that control the behavior of the application. Each task in the application is a form with fields to be filled. Filling of fields causes different events to fire. When certain combinations of fields are filled, a given task can be completed. For example, a form/task that handles money transfer between funds needs at least 3 fields to complete the transaction – a fund to buy, a fund to sell, and the amount of money that should be transferred. For example, "Transfer *$3000* from the *Growth Fund* to *Income* fund".

Prompt markup defines the prompts used in the application. Every form defined has an associated prompts file that contains the prompts to be played when a particular field in the form is filled, when combinations of fields are filled, when the form is completed or when there are errors in field values. The prompts can use dynamic data and support rich speech user interface.

Event Handlers execute in response to an event that fires when a field or combinations of fields are filled. The default handlers in the CIM play associated prompts from the corresponding prompts file. These defaults can be overridden by the application to execute task-specific logic, like validating field values.

Every turn of user input is interpreted and scored by a form and field scoring algorithm to determine the task associated with the user's request. Then the associated form gets focus and a series of events are called to prompt the user via successive turns to provide all field values to complete the form.

3 Dialog and Prompt Features

The CIM has several default features to promote rich dialog and speech user interface best practices.

3.1 Prompt Features

The CIM provides several prompt features such as *self-revealing help* to handle user silences or low confidence recognitions, the ability to specify *form and field specific help prompts* for help requests, the ability to *expire* a prompt after a specific number of turns and the ability to *rotate* and *randomize* prompts to avoid repetition. It also enables *intelligent repeating* of informative prompts for user requests of *repeat* instead of blindly repeating the last prompt.

Prompts associated with a field or an event can be grouped together to facilitate playing of self-revealing help prompts over successive turns of incorrect input or no input. Any prompt or group can also optionally evaluate a boolean expression containing field combinations that need to be filled in order for that prompt to be played. This feature can be used to reinforce the information collected so far to the user during the next prompt. For example, when the user has only provided a "BUYFUND", the prompt may be *"How much would you like to transfer to the Growth Fund"*. If the user has provided both a BUYFUND and SELLFUND, the prompt may be altered to be *"How much would you like to transfer from the Income Fund to the Growth Fund?"*

The CIM provides a powerful feature that allows dynamic data elements from a data structure containing back-end data to be referenced from within the prompt. This is done via the use of *macros*. *Substitution Macros* allow the prompt to reference a single dynamic data element. *List Expansion Macros* allow the prompt to reference one dimensional and multi-dimensional data structures by expanding them in a nested manner during run-time. They allow repeated extraction of data elements from a data node and can also provide a count of the number of elements in the list. Parameters to the macro include a pointer to the data node, a prompt prefix such has "You have (list-length) funds" where (list-length) evaluates to the number of elements in the data node, and the repeating portion of the prompt, etc. It also provides the ability to specify final conjunctions, and makes intelligent use of "is" or "are" as needed.

For example, a one-dimensional list expansion such as "*You have (2) funds: ($3000 in Growth Fund) 'and' ($2000 in Income Fund).*" could be specified by providing a pointer to a data node containing fund names and balances in each. In a similar manner, the CIM can process multi-dimensional nested expansions such as, "*Your have (2) matching (American) flights: one at (7.20 AM) 'and' one at (8.30 AM) 'and' 3 matching (Delta) flights: one at (1.20 PM), one at (4.30 PM) and one at (9 PM).*" where the airline names and timings are the two dimensions.

3.2 Handling Task Switching and Inheritance

Handling dynamic task switching (or digressions) and return effectively is very important for a good user experience in mixed-initiative systems. In such applications, callers often switch tasks to get some information before responding to the current prompt. This behavior is similar to how users might interact with live agents. For example,

System: Fund Transfer. Which fund would you like to sell?
User: Which funds do I own (→ *Dynamic task switch from Fund Transfer to Fund Information*)
System: You own 3 funds, Growth Fund, Income Fund and Stable Fund. Which Fund would you like to sell (→ *return to Fund Transfer and re-prompt*)

In addition to facilitating dynamic task switch and return, the CIM also provides the ability to confirm the task switch with the user before switching. The prompt that is used for this confirmation can be customized based on the current and target tasks.

The CIM also provides the ability to pre-fill certain fields in a target form from values provided by the user for the source form. In order to provide the user an experience of a seamless transition between forms of the application, the CIM maintains dialog history of previous user interactions. For example,

System: You have $5000 in the Growth Fund. What else can I help you with?
User: How about the Income Fund? *(→ Assumes Balance Request based on history).*
System: You have $2000 in the Income Fund.
User: I'd like to sell that. *(→ Fund Transfer task inherits the Income Fund as SELLFUND).*
System: Fund Transfer. Selling the Income fund. Which fund would you like to buy?

3.3 Handling Confirmations

Confirmations in conversational mixed-initiative systems handle a wider variety of input than a straightforward *yes* or *no* such as implied yes/no, yes/no with additional information, yes/no with task switch or any of the above without yes or no being specifically present in the input. The CIM is designed to handle all of the above cases in a customizable manner. It can also handle nested confirmations and provides the ability to have self-revealing help within the confirmation.

4 Summary

The Conversational Interaction Manager provides the a flexible and customizable mixed-initiative application development framework with extensive and rich default behavior satisfying the needs of real-world sophisticated applications.

5 References

"A Statistical Mixed-initiative Dialog System for 401k Management", Rajesh Balchandran, Linda Boyer, Mark Epstein et al, *Proceedings Human Language Technologies Conference (HLT-2002).*