A Multi-modal Dialog System for Business Transactions

Joyce Chai, Sylvie Levesque, Margorzata Budzikowska, Veronika Horvath, Nanda Kambhatla, Nicolas Nicolov and Wlodek Zadrozny

IBM T. J. Watson Research Center
30 Saw Mill River Road, Hawthorne, NY 10532, USA
{jchai, sm1, veronika, nanda, nicolas, wlodz}@us.ibm.com, slevesqu@ca.ibm.com

ABSTRACT

This paper presents a general framework for conversational agents for business applications that supports multi-channel, multi-modal interactions through the use of a channel and modality independent Dialog Move Markup Language. In particular, we describe a prototype system as an instantiation of a general dialog architecture that supports web-based interaction through a combination of modalities such as natural language dialog and user interface components. User studies have revealed that the prototype system has enhanced user experience in an online shopping environment by significantly reducing the length of interactions in terms of time and the number of clicks. Furthermore, the success in extending the general architecture to a prototype system demonstrates the applicability and potentiality of such framework in business applications.

1. INTRODUCTION

Advances in e-commerce applications have created an increased demand for systems that can provide fast, efficient and user-friendly information access and timely completion of business transactions. Such systems have to accommodate varying customer needs and business requirements, and more importantly, natural, intuitive means of interaction with the user. Traditional structure based or menu-driven systems tend to overwhelm and frustrate users with lengthy, rigid interactions. We believe natural dialog is an effective medium for understanding user requests or intentions, for providing advice and recommendations, and for helping users with different shopping behaviors and expectations to accomplish business transactions. Furthermore, multi-channel, multi-modal natural dialog is particularly appealing and useful for communicating with varied devices/channels (e.g. web, telephone, PDA, cellular phone, etc.) and modalities (e.g. speech, text, graphics, etc).

We are currently working on a generic framework for building conversational dialog agents for business applications, where users can converse with the agents using any channel of interaction or modalities. Conversational dialog agents are automated software agents that can participate fully in natural dialog [1] and whose internal state may include beliefs, desires, and intentions (BDI models; [2,3,4]). Examples of conversational agents include natural language dialog based telephony banking and stock trading systems [5], a planning system for disasters [6], email checking system [7]. We have implemented a web-based natural language dialog enabled prototype system (Natural Language Sales Assistant) as an instantiation of this framework [8]. In this paper, we will describe the general framework and the Dialog Move Markup Language, followed by the basic structure and functionalities of the Natural Language Sales Assistant (NLSA) and results from user studies.

2. A FRAMEWORK FOR MULTIMODAL MULTI-CHANNEL INTERACTION

The general architecture relies on a clear separation between Presentation Managers (PMs) and Conversational Dialog Manager (CDM). This architecture supports multi-channel, multi-modal interaction through the use of several presentation managers, one for each channel of interaction (e.g., web, telephone, PDA, cellular phone). A channel can support one or more modalities (e.g., speech, text, forms, GUI widgets, graphics).

![Diagram of the multi-channel multimodal system](image)

Figure 1: General architecture for the multi-channel multi-modal dialog systems

Each presentation manager interacts with the user through a specific channel. A presentation manager uses a suite of APIs to access services (e.g. NLP, NLG, etc.) to communicate with the user through the appropriate modalities over the given channel. The presentation manager communicates the user intentions (on his/her behalf) to the conversational dialog manager in a common interaction format, the Dialog Moves Markup Language (DMML). The Dialog Manager uses a suite of APIs to revise its discourse model, apply business logic and execute business transactions, possibly through an Action Manager (AM). The dialog manager generates DMML messages to communicate results to the presentation manager and to communicate its intention to pursue or trigger interaction with the user.

2.1. DMML

The Dialog Move Markup Language (DMML) [9], inspired by speech acts theory, is used by the conversational agents to communicate intentions independently of the specific features of particular channels, modalities or applications. DMML is an
application of the eXtensible Markup Language (XML) [10], a standard for document and message markup.

The main elements of DMML are dialog moves and dialog profiles. A dialog move by agent A to agent B represents a set of dialog acts by agent A directed towards agent B, with the intention of changing agent B’s model of the world or to convince agent B to take action, based on agent B revised model of the world. Table 1 gives a list of the dialog and communicative acts constructs we found most useful for dialog management in an e-commerce environment. The basic set of communicative acts covered in DMML is easily extendable. Dialog profiles enable agents to establish the parameters of a conversation by specifying constraints on the environment (e.g., restricting the set of admissible dialog moves). Dialog profiles can be communicated at any point in the dialog and span more than one dialog move. They set the global context for subsequent moves.

<table>
<thead>
<tr>
<th>Dialog Act</th>
<th>Communicative Act</th>
<th>Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>assertion</td>
<td>greeting, warning, reminder, thanks, welcome, offer.</td>
<td>agent A wants agent B to know about C. Agent B is not required to answer the volunteered information.</td>
</tr>
<tr>
<td>command</td>
<td>help, exit, cancel, operator.</td>
<td>agent A wants agent B to unconditionally execute an action C.</td>
</tr>
<tr>
<td>request</td>
<td>information, confirmation, clarification, identification, action execution notification.</td>
<td>agent A wants agent B to do C.</td>
</tr>
<tr>
<td>response</td>
<td>notification, clarification, confirmation, action results, information, answer_list, description, explanation.</td>
<td>agent B wants agent A to know C as the result/ consequence of a previous request from agent A.</td>
</tr>
</tbody>
</table>

Table 1: DMML structure for dialog moves.

DMML has no domain or application specific markup tags. All application and domain specific tags are encapsulated by the DMML markup itself. This allows building sophisticated natural language dialog systems across e-commerce applications in a cost-efficient way through reuse as it separates linguistic and transaction knowledge levels. DMML messages can embed arbitrary XML content. This content includes xml-schemas, thus allowing agents to communicate expectations and constraints on requested data and the content of further messages. An example of a DMML message is given in the following section.

3.1. Data Management

3. NATURAL LANGUAGE SALES ASSISTANT (NLSA)

NLSA supports web-based interactions with modalities such as text, forms, and standard user interface components. It allows users to specify what they are looking for either in their own language or through radio buttons. Through a sequence of conversations that based on both natural language dialog and interactions through user interface components, the system will guide users to relevant IBM ThinkPads. The NLSA consists of two subsystems: data management subsystem and online interaction subsystem as shown in Figure 2.
Each entry in the Concept Base contains a list of synonyms and implicational keywords associated with the concept (<SYN> and <MP> elements), a set of natural language expressions used by the system for response generation (for example, <NOVICE> and <EXPERT> elements are questions prompted to the user to solicit information concerning this concept), as well as a business definition for the concept in terms of product specifications (<DEF> element). In addition, concepts may also contain feature attributes such as “scaleable” or “ambiguous.” Concepts can be as concrete as entities such as products and accessories (e.g., “DVD”), or as abstract as attributes (e.g., “LIGHT-WEIGHT”) or purposes (e.g., “BUSINESS-USE”).

All information in the Concept Base can potentially be updated semi-automatically through a set of tools by developers who are familiar with a particular domain and application. We are currently working on developing tools that will allow the customization to different domains to be limited to pure knowledge updating without modifying the Online Interaction Subsystem.

3.2. Online Interaction

The Online Interaction Subsystem consists of three major modules: Presentation Manager (PM) which is responsible for separating content from presentation, Dialog Manager (DM) which employs a state-based approach for content selection in the dialog arrangement, and Action Manager (AM) which accomplishes back-end database access, information retrieval and business transactions.

In particular, the Presentation Manager utilizes a natural language noun phrase parser to transform user natural language query into a logic form and sends an instance of Communicative acts with embedded logic form to the Dialog Manager (see the following example). Based on the response content sent by the Dialog Manager (in the form of Communicative acts), the Presentation Manager applies an Explanation Model to generate natural language explanation and a Response Generator to generate specific presentations based on appropriate modalities such as display tables, natural language output, GUI components, etc.

The Dialog Manager formulates action plans in the form of Action Specs for the Action Manager to accomplish back-end business oriented tasks and arranges follow-up dialog based on action results returned by the Action Manager and current dialog state. The Dialog Manager applies a state-based approach to formulate responses to the user depending on both the short-term conversational discourse history and long-term user history. The system applies a mixed initiative dialog strategy by giving the initiative to the user at the beginning of the interaction (where users can specify what they are looking for) and then dynamically switching the initiative between the system and the user based on the different dialog states. A dialog state is a representation of the current status of dialog that takes into consideration the current utterance, the current system response and the interaction history. A particular dialog state is determined based on the following factors: any user preference from other UI components, any ambiguity in the user natural language input, any concept or dialog act identified from the current utterance, any conflict constraints from a history of interactions, any action results from the Action Manager (such as the number of products recommended). At each turn of interaction, an explicit confirmation strategy is applied. A feedback is generated to the user as to how much the system has understood so far and what the system limitation is in order to encourage the user to be more co-operative and to phrase the request under this limitation.

Communicative acts

Based on the current dialog state, the Dialog Manager arranges the responses through different communicative acts. Figure 3 shows how some basic DMML communicative acts for response and request have been extended to suit the specific needs of the NLSA.

Communicative Acts for Response

<table>
<thead>
<tr>
<th>Explanation</th>
<th>Clarification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation</td>
<td>Ambiguity-resolution</td>
</tr>
<tr>
<td>&lt;more-information&gt;</td>
<td>&lt;answers-to-most discriminating-question&gt;</td>
</tr>
</tbody>
</table>

Communicative Acts for Request

<table>
<thead>
<tr>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;confirmation&gt;</td>
</tr>
</tbody>
</table>

Figure 3: Communicative Acts for NLSA

The following example of a DMML message from the Presentation Manager to the Dialog Manager corresponds to a user request for “nice travel companion between $2000-3000 with at least 400MHz Pentium processor”. The DMML markup itself is in shown in bold and the application specific content is shown in normal and italic fonts.

```
<dmlm>
  <move from="PM" to="DM" id="searchRequest" moveid="request_1">
    <request type="information">
      <xmlns:id="searchRequest">
        <nm><fps value="for"/><np>
          <head><specs concept="PORTABILITY">
            <attribute><specs type="CPU_BRAND">Pentium</specs></attribute>
          </head>
        </np></fps>
      </nm>
    </request type="information">
    <set xmlns="searchRequest">
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="CPU_SPEED">gt400</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="BASE_PRICE">lt3000</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="BASE_PRICE">gt2000</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="CORE_SPEED">gt400</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="CORE_SPEED">lt3000</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="CORE_SPEED">lt2000</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="MEMORY">gt128</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="MEMORY">lt128</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">gt1024</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt1024</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt512</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt256</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt128</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt64</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt32</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt16</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt8</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt4</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt2</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt1</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt0.5</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">lt0.25</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE" model="closed"/>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="MEMORY">gt128</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="MEMORY">lt128</specs></attribute>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="MEMORY" model="closed"/>
        </head>
      </nm><fps value="with"/>
      <nm><fps value="for"/><np>
        <head><specs concept="PORTABILITY">
          <attribute><specs type="HARD_DRIVE">gt1024</specs></attribute>
        </head>
      </nm><fps value="with"/>
The first part of the request application content (in normal font) is a XML fragment containing the logical form derived from the user input by the presentation manager. The second part of the application content (shown in italics) is an xml-schema that communicates to the dialog manager the constraints on the structure of data that the presentation manager expects to receive as an answer.

3.3. User Studies

Several user studies have been conducted to objectively evaluate the usability of the NLSA to better understand user needs and system limitations [8]. The result of a comparative study (comparing NLSA with a fully developed menu-driven system) shows that users (seventeen testing subjects) prefer the natural language dialog enabled navigation two to one over the menu driven navigation. In addition, the study confirmed the efficiency of using natural language dialog in terms of the number of clicks and the amount of time required to obtain the relevant information. The average number of clicks used in the natural language system was reduced by 63.2% and the average time was reduced by 33.3%. Furthermore, we found that sophisticated dialog management is more important than the ability to handle complex natural language sentences. Analysis of the user queries (the average length of a user query was 5.31 words long, with a standard deviation of 2.62; 85% of input are noun phrases) reveals the brevity and relative linguistic simplicity of their input; hence, simple parsing techniques seem adequate to extract the necessary meaning from user input.

We have also learned that in order to improve the functionality of an e-business site, the natural language dialog navigation and the menu driven navigation should be combined to meet users’ different needs. While the menu driven approach can provide choices for the user to browse around or learn some additional information, the natural language dialog provides the efficiency, flexibility and natural touch to the users’ online experience.

4. CONCLUSION

Dialog is an effective medium for understanding user intentions and providing additional information and recommendations in e-commerce applications. With many devices and communication channels available, it is desirable for a system to provide consistent interactions despite discrepancies in modalities and channels. This paper presents our approach in designing such systems. In particular, our experience in extending the general architecture to a successful web-based prototype system manifests the applicability and potentiality of the proposed framework in business applications. Furthermore, we have learned that not only DMML based content communication, but also channel/modality specific strategies are important in such context. The web channel introduces a new dimension for information presentation. Through a combination of UI components (such as radio buttons, forms, links, etc.) and the natural language dialog, more information can be communicated in comparison to a traditional spoken dialog system. Thus, with web specific presentation strategies and dialog strategies, the user can get instructions, examples, explanations, and ideas of limitations of the system and eventually reach the final goal with reduced number of turns (interactions).

5. REFERENCES