Ontology-Based Utterance Interpretation for Intelligent Conversational Interfaces

Emerson Cabrera Paraiso
Pontifical Catholic University of Parana
Av. Imaculada Conceição, 1155
Curitiba, PR - Brazil
+ 55 41 3271-1676
paraiso@ppgia.pucpr.br

ABSTRACT
In this paper, we present an ontology-based utterance interpretation mechanism for intelligent conversational interfaces. We describe how this mechanism was embedded in a conversational interface applied to personal assistant agents. The main goal of such approach is to offer a system capable of performing tasks through an intuitive interface, allowing experienced and less experienced users to interact with it in an easy and comfortable way. In this mechanism, ontologies are used for syntactic and semantic interpretation. We present how to design an ontology for semantic interpretation and how the interpretation process use it for semantic analysis.

Categories and Subject Descriptors
H.5.2 [User Interface]: Natural language.

General Terms

Keywords
Speech interface, dialog systems, assistant agents, ontology, conversational interfaces.

1. INTRODUCTION
In any software application, an appropriate user interface is crucial. Traditionally, developers propose graphics-oriented interfaces involving the use of menus, sub-menus, dialogue-boxes, and so on. Often this approach is inappropriate or at least not very appealing, leading to an interaction of poor quality [11]. To produce better interfaces, from the user interaction point of view, we are applying conversational interfaces. Conversational interfaces as defined by Kölzer [9], let users state what they want in their own terms, just as they would do, speaking to another person. Such interface allows the application handle real dialogues with users. One of the most difficult task in implementing a conversational interface is to interpret utterances and understand their meaning. To do that, we are using ontologies. The ontologies play a key role at the semantic interpretation time since the meaning of utterances can be inferred by looking for concepts and their attributes. The use of ontologies for representing domain knowledge and for supporting reasoning is becoming widespread. The ontologies however, may also be used for facilitating the interaction between user and the system. The concepts and their properties are organized to map the world but also to help processing natural language. In this paper, we present an ontology-based utterance interpretation for intelligent conversational interfaces. We have been applying conversational interfaces for personal assistant agents with success. A personal assistant (PA) is a specialized artificial agent that helps human users to do their daily work. Personal assistants help users reducing the ever-growing load of information, events and various commitments they need to handle, for instance by learning how to organize and keep track of relevant items [15].

The approach of utterance interpretation presented in this paper was used to implement SpeechPA: an ontology-based speech interface for personal assistants. We applied SpeechPA in two different projects: one in the domain of computer supported cooperative work (CSCW) [11] and the other in the domain of e-government (a multi-agent system to support users (civil servants) advising citizens -see [12] and [17] for details).

This paper begins by presenting the SpeechPA architecture. We then show how to design an ontology for semantic interpretation and how the interpretation process use it for semantic analysis. We give some examples on how the mechanism works. Finally, we offer a conclusion and indicate some perspectives.

2. SPEECHPA: AN OVERVIEW
SpeechPA is an ontology-based speech interface for PA. Such interface allows the PA handle real dialogues with users. The design and implementation of such interface is a difficult task that involves many different modules: dialogue controllers, natural language parsers, speech recognizers and synthesizers, knowledge manipulators, to list a few.

Figure 1 shows the major components of our architecture. It has three parts: (i) graphical and speech user interface (GSUI) modules; (ii) linguistic modules; and (iii) agency modules. The process starts by capturing the utterances using a commercial automatic speech recognition engine that returns the recognized result for each word. The Utterance Capturing module concatenates all the words forming an utterance. A process running independently analyzes each utterance. Due to local noise interference or bad pronunciation, the utterance may be lexically and/or syntactically different from the words actually said. Initially, we are using the utterance as it is, extracting a list of known disfluencies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC’08, March 16-20, 2008, Fortaleza, Ceará, Brazil.
Copyright 2008 ACM 978-1-59593-753-7/08/0003…$5.00.
Interpreting an utterance is done in two steps: (i) parsing and syntactic analysis; and (ii) ontology application. The results are sent to the dialogue manager continuously, or back to the user when they do not make sense. The parsing algorithm works top-down. It replaces each utterance stem with its syntactic category (verb, noun, adverb, etc) with the help of a lexicon file and a set of grammar rules. The grammar rules were divided in order to classify an utterance into one of three categories: order, question or answer. If a sentence is not well formed, or if it is out of the domain, then it is classified as a nonsensical utterance. In such a case, the user is invited to reformulate his sentence. Nonsensical utterances occur rarely since our system tries to act with minimal information, but nevertheless may occur.

The mixed-initiative and task-oriented dialogue mechanism is coordinated by the Dialogue Manager module. Each dialogue session is conducted as a task with sub-tasks and can handle several tasks simultaneously.

When the user requests an action, the Dialogue Manager tries to execute it, creating a task that is dispatched by the Action Looping module. However, if the initial utterance lacks crucial information—e.g., an action parameter—the Dialogue Manager starts sub-tasks to complete the action list, asking additional information from the user. To do that, it uses an action library. Once the action list is complete, the PA executes it, with eventual support from other agents.

In SpeechPA ontologies are written in OWL [10]. We use an ad-hoc mechanism for reasoning [1].

Although talking is a privileged mode, the agent interface is multimodal. Thus, the Action Looping is responsible for merging all modalities (e.g., button click and speech).

The real improvement in this conversation interface is the way knowledge is handled and used to interpret utterances. The next sections give more details.

3. SOME ISSUES ON WRITING DOMAIN ONTOLOGIES FOR SEMANTIC INTERPRETATION

The design of an ontology depends on its intended function (the reader is referred to [13] and [4] for a quick review of the domain). Eriksson in [3] addresses the issues of designing ontologies for dialogue interaction and information extraction. In her work, she explores the requirements of an ontology specially designed for dialogue systems. In [3], Eriksson states that ontologies provide a common vocabulary that can be used to state facts and to formulate questions about the domain. Dzikovska et al. [2] present a method for customizing a broad-coverage parser to different domains by maintaining two ontologies (one that is generalized for language representation, and another that is customized to the domain), and for defining mappings between them.

Many researchers have applied semantic-driven approaches (e.g. [2], searching keywords or phrases in the utterance (user statement). Our approach to semantic interpretation however, is based on the notion that the meaning of utterances can be inferred by looking for concepts and their attributes (more details in section 4).

To interpret users’ statements and to manage the dialogue between user and the PA, some simple rules should be respected when writing the ontology.

Figure 2: An excerpt of the domain ontology

The domain ontology shown in Figure 2 was elaborated following the principles described in the next sections and it is an excerpt of the one built in a project related to knowledge management in R&D projects (see [10] for further details).

3.1 Definition of Concepts and Their Relations

Concepts and their relations (binaries) are the basic elements that constitute an ontology. The use of hyponymy and hypernymy to link concepts (is-a relation) and meronymy to describe dependency (has-a) may help to interpret incomplete or unexpected statements. For instance, the user demands to the PA: List all articles on agents. According to the ontology (Figure 2), finding conference articles or journal articles in the statement could be easier to interpret. However, thanks to the hyponymy relation between articles and conference/journal, a formal representation would be obtained as well. Acting in the same way, the reference resolution may be guaranteed as shown in the following short dialogue fragment:

PA: Which document do you want to list?
USR: Articles.
PA: Do you mean conference articles or journal articles?

---

1 We intentionally reduced the ontology to a bare minimum, extracting some concepts, properties, relations and labels.
The semantic analyzer identified articles as a hyponym of documents (is-a relation).

It is also helpful to define a list of synonyms to each concept.

3.2 Definition of Properties
   To describe each property of a concept, one should define its type, a list of synonyms, its cardinality and a domain restriction. Domain restriction should be one of the following:
   - Time: for temporal attributes (e.g. appointment: time, duration or hour);
   - Space: for geographic localization;
   - People: for describing people;
   - General: for others attributes.
   A restriction is especially useful when interpreting questions (see examples in section 4).

3.3 Definition of Actions
   An action is a token used to describe operational tasks applied to a concept. An action token is used by service agents to identify whether it is related to its domain or skills or not. Each action token has a skill associated. A skill can be expressed as a list built respecting the BNF formula. This formula is a list built respecting the BNF formal representation is a well-formed computational formula.

3.4 Definition of Multiple Instances
   Since ontologies should be able to incorporate several views of a domain, e.g. user and system, or several different information sources, as advocate Eriksson [3], multiple instantiation is important. Multiple instantiation is useful for defining the list of tasks in the task ontology, as well.

The resultant ontology contains domain information but also lexical and synonyms information. To avoid any misunderstanding, it is important to highlight that the main ontology structure (concepts and their relations) construction is guided by the domain application and not by its utilization in the semantic interpretation system.

4. SEMANTIC INTERPRETATION PROCESS
   The process of interpreting a user statement is carried out in two steps: parsing and syntactic analysis; and semantic interpretation (domain ontology application).

In order to reduce the interaction and, consequently avoid wasting time, we limited the space of dialogue utterances to directive speech act classes [16]—inform, request, or answer—since such classes define the type of expected utterances in a master-slave relationship. A speech act is an act that a speaker performs when making an utterance. The idea is to do something by looking for the related acts in the utterance. The strategy of treating directive speech acts reduces the number of turn-takings since some speech acts, like acknowledgement acts (“Thank you” or “Have a nice trip”) will not be used by the PA. In our applications, a typical utterance could be: “Look for a document on agents in the database.” According to our taxonomy this is an order utterance and can be processed by the grammar rules. If a sentence is not well formed, or if it is out of the domain, then it is classified as a nonsensical utterance. In such a case, the user is invited to reformulate her sentence.

The approach to semantic interpretation presented here is based on the notion that the meaning of user statements can be inferred by looking for concepts and their attributes. More precisely, the module responsible for applying the ontology to the statement searches for domain concepts and the list of verbs that indicates the task to be executed. The corresponding keywords are concepts of the ontology directly related to a list of actions. We believe that this approach is ideal for applications where the domain is well known and restricted. In contrast, statistical models as proposed by [6] bring the analysis to the parser level, leaving useful domain information out of the process.

After interpreting a user entry, we have a formal representation of it. Thus, this formal representation is the semantic analysis result. The formal representation is a well-formed computational formula. This formula is a list built respecting the BNF specification shown in Table 1.

To exemplify the semantic interpretation process, let us introduce the following entry:

**USR:** Do I have a meeting with Mike in my office?

Firstly, the statement is parsed and a syntactic tree is obtained. To interpret the given utterance, first the parser checks the context of the input, verifying that it is a question related to the domain. To do so, it uses the domain ontology and the lexicon. The lexicon contains thousands of words extracted from WordNet [4] and enriched with the list of all concepts and attributes of the ontology. Additional information is collected, by identifying
relations between pairs of words, based on a link grammar adapted from Link Parser [6].

The statement is classified as a question by the parser so, to be able to follow the conversation, the PA must answer the given question. In order to answer the question, the PA built the formal representation of it, before send it to the concerned service agent:

\[
\text{(list (Meeting (:participant "Mike") (:place "office")))}
\]

This list was obtained from an algorithm that searches for domain concepts. The verb-object link between have and meeting points the phrase object. The algorithm searches an equivalent concept in the domain ontology (Figure 2). It finds Meeting, a hyperonym of Appointment. Comparisons between words and concepts (and their synonyms) are made with help of a word similarities mechanism.

After selecting the candidate concept, the algorithm starts a process to try to fill up all concept properties. The Meeting concept as six properties: duration, date, place, time, participant and description. Note that the parser identified two constituents, each constituent have a potential concept property on it. Each instance of a concept is kept in memory for future use (stack of instances). For this type of question, the action list is systematically used.

The process of interpreting a direct question (such as what, where, when, who, etc.) is particular, since the dialogue manager does not search for an action to be executed. Instead, it searches for an object desired by the user. For instance, assume the question below:

USR: Where is the meeting with Paul?

The information which the user looks for is indirectly referenced. When the user uses where, she actually wants to know where the meeting will take place. In this case, the semantic analyzer will use the restrictions of each found concept’s attribute. The attribute place has the restriction space, indicating the geographic location of the meeting. Thus, the formal representation may be obtained:

\[
\text{(list (place (Meeting (:participant "Paul")))})
\]

Once the formal representation has been obtained, in all cases, the PA tries to find an answer, with help of others agents if needed. This list-based representation is used by agents to infer the answers. They have inference machine capable of doing that.

To interpret the statement “Send an email to Mike”, the extract of ontology shown in Figure 4 is used.

Following the process just explained, the formal representation can be obtained:

\[
\text{(send (e-message (:receiver "Mike")))}
\]

The ontologies are also used to solve another important problem on dialogue management: the reference. Reference resolution is a complex problem been object of active research on linguistics (see [14] and [7] for details). Reference in dialogue systems is the study of how objects from the real world are presented inside the utterances, how they should be stored and how they can be used for keeping the conversation context. The simplest form of reference appears when in a statement is found a new object or a past introduced object, in the context. To better understand, let’s consider the excerpt of dialogue between a user and his PA shown in Table 2.

<table>
<thead>
<tr>
<th>Table 2: An excerpt of dialogue between a user and his PA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USR (1):</strong> Give me the list of articles written by Mike Palmer.</td>
</tr>
<tr>
<td><strong>PA (2):</strong> [PA perform the task: list articles from Mike Palmer]</td>
</tr>
<tr>
<td><strong>USR (3):</strong> Is he a participant of the project?</td>
</tr>
<tr>
<td><strong>PA (4):</strong> Yes</td>
</tr>
<tr>
<td><strong>USR (5):</strong> Do you known his telephone number?</td>
</tr>
<tr>
<td><strong>PA (6):</strong> Yes. The telephone number is: 041 3271 1515</td>
</tr>
<tr>
<td><strong>USR (7):</strong> and his address?</td>
</tr>
</tbody>
</table>

In the first statement (1), the user cites the name of Mike Palmer for the first time. Whether the system wants to keep the context of the conversion, it should store that information. Then, in the next statement (3), the user cites Mike again, through the pronoun he. This phenomenon is known as anaphora, e.g., the fact of to reference an entity already introduced in the dialogue.

Another well known phenomenon in a dialogue is the occurrence of an ellipsis. The detection of an ellipsis is done by analyzing a statement, that at a first glance, is incomplete, but where the “missing parts” were presented in previous statements. In the dialogue, at the statement (5) the user demands the telephone number of Mike and in the statement (7) he adds a demand of his address. Here, the statement (7) can be interpreted as a complement of the statement (5).

In the actual version of our system, we only treat the presence of an anaphora. To do so, after interpreting each user statement, the dialogue manager creates or updates the instances used to interpret the given statement. For example, given the short dialogue below:

USR (1): What is the email address of Mary?
PA (2): The email is: mary@company.com.
USR (3): Send an email to her.

By interpreting the statement (1), the semantic analyzer builds the formal representation:

\[
\text{(list (address (AddressBook (:name "Mary"))])}
\]

At the same time, the dialogue manager stores in a stack of instances of concepts, the instance below:

\[
\text{(e-message (:receiver "Mary") (:subject nil) (:content nil))}
\]
The dialogue manager may now to solve the reference to Mary in (3) thanks to the instances stored in the stack, since probably the object just referred appeared in the latest statements. The same occurs in the next short dialogue:

USR (1): What is the starting time of the meeting with Mike?
PA (2): Starting time is: 14h00
USR (3): Where is it planned to be?

The presence of the pronoun it in (3), leads the dialogue manager to search to an instance of a compatible concept. In the top of the stack of instances it will find the instance:

\[
\text{(Meeting} \ (:\text{date} \ "17-05-2006")
\quad (:\text{time} \ "14h00")
\quad (:\text{place} \ "office")
\quad (:\text{duration} \ "1h00")
\quad (:\text{participant} \ "Mike")
\quad (:\text{description} \ nil))
\]

The response is then posted to the user since all needed information is already known.

5. CONCLUSIONS

In this paper we presented how ontologies can be used to support intelligent conversational interfaces. Using ontologies as the support to implement conversational interfaces, one can develop intelligent user interfaces through which unsophisticated users can have their requests treated in a fast and useful manner, guarantying predictability. Impossible requests, such as those out of context, are easily handled since the system uses a list of applicable tasks described in the ontology.

We are currently working in an intelligent agent architecture for on-line assistance. The intent of this architecture is to provide a framework to develop web-based intelligent agents that give useful information on a specific domain. In this context, ontologies are used to interpret users’ demands and also to generate natural language responses.

6. REFERENCES


