An Event Driven Model for Dialogue Systems

Kuansan Wang

Speech Technology Group, Microsoft Research
One Microsoft Way, Redmond, WA 98052

ABSTRACT

This paper reports our progress in building a mixed initiative, goal-oriented dialogue system for human machine interactions. The dialogue model embraces the spirits of the so-called plan-based approach in that the dialogue flow is not statically authored but dynamically generated by the system as a natural outcome of the semantic evaluation process. With multimodal applications in mind, the dialogue system is designed in an event driven architecture that is commonly seen in the core of a graphical user interface (GUI) environment. In the same manner that GUI events are handled by graphical objects, the proposed dialogue model assigns dialogue events to semantic objects that encapsulate the knowledge for handling events under various discourse contexts. Currently, we have found that four types of events, namely, dialogue object instantiation, semantic evaluation, dialogue repair, and discourse binding, are sufficient for a wide range of applications.

1. INTRODUCTION

As more and more information can be accessed through automatic agents, the quality of human machine interaction has become a critical issue for system usability. In most cases, it is unlikely that a user can access desired information in a single query. More often than not, the query might be ambiguous, incomplete, or even incoherent with respect to the interaction history. Even if the request is well formed, precise, and perfectly recognized by the system, it is quite common that the size of the legitimate answers is so huge that the system has to consult with the user to reach a useful and comprehensible response. All these require the system to engage in a dialogue with its human user to progressively refine the goal and complete the task. Accordingly, the dialogue system plays a forefront role in the human machine interactions.

With the advancements of computing technologies, it is reasonable to believe that the user expectations on dialogue system will quickly evolve from "just getting things done" to "getting things done intelligently." How to equip a dialogue system with sufficient amount of intelligence has been an actively studied research topic. One popular model for intelligent dialogue systems can be lumped under the name of plan-based approach [1-3]. The key belief here is that, ultimately, the conviviality between the interlocutors should emerge naturally from the interactions rather than be dictated and hand coded *a priori* by the system's designers. In other words, an intelligent dialogue system should be capable of managing the contextual exchanges with its user and inferring the proper course of actions *dynamically* based on the semantics of the discourse. One obvious benefit with a plan-based system is that the application designers are alleviated from the burden of having to exhaustively enumerate all the possible dialogue flows and program the system's responses for all of them in a painstakingly step-by-step fashion.

Many dialogue systems have employed speech as the basic communicative medium. For telephony applications, where speech is arguably the most compelling modality for communication, such a speech centric view is quite adequate. However, in a computer environment in which a multitude of communication channels are usually available to the user, the realization of a dialogue system must be augmented so that the various modalities (e.g. speech, gesture, hand writing, etc.) can be seamlessly integrated into forming a complementary environment that enhances user experience.

In this article, we report our progress in implementing a computational architecture for multimodal dialogue systems. The proposed architecture is based on the event model commonly seen in a GUI environment for ease of integration with the existing interface paradigm. The strength of the plan-based approach, namely, to infer a proper course of actions based on logical reasoning, is embraced in the proposed model. In general, the proposed model views the dialogue as an integrated part of the discourse semantic evaluation process, which all the dialogue actions are natural outcomes of. The proposed dialogue management system is designed to function in conjunction with a signal understanding unit as shown in Fig. 1. The understanding unit processes the input signal, extracts and translates the embedded message into a semantic representation based on a set of principles, which we call the *semantic model*, and the discourse context provided by the dialogue management system. The problem of signal understanding and the mechanisms we use to tackle the problem are further elaborated in Sec. 2. The semantic representation is then passed on to the dialogue management system for semantic evaluation. The evaluation process is aided by a set of event handlers that contain domain specific knowledge pertinent to achieving the dialogue goal. The behavior model guides the dialogue management system to synthesize proper responses in case any interactions with the human user are needed. As we focus on the aspects of automatic dialogue flow generation in this article, behavior models are out of the scope for our discussion

2. SIGNAL UNDERSTANDING

It is assumed that the meaning of a single is carried by a series of semantic objects in the same manner that a sentence is composed of words. With such a view, it is appropriate to treat the problem of signal understanding as a pattern recognition problem, in which the patterns to be recognized are the semantic objects. The task of designing a signal understanding

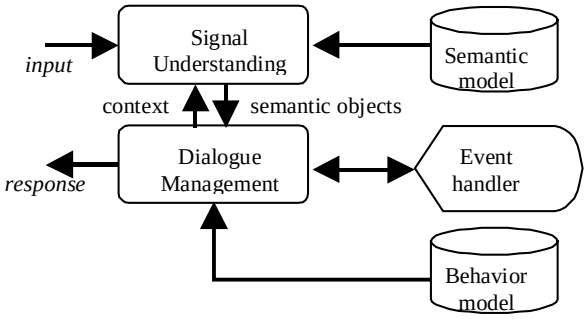

Figure 1: A system diagram of the event based dialogue system.

system, therefore, is to come up with a decision rule such that the semantic objects are identified in an optimal way.

As common to many pattern recognition problems, it is usually more practical to embrace the principle of self-diversification and construct the patterns from smaller and more manageable units. In automatic speech recognition, for instance, a speech utterance is usually thought to be composed of a sequence of subword units (e.g., phones or senons) that serve as the building blocks of a language. For an understanding system, the same principle may also apply, but there exist distinctive challenges. First, even in the speech only case, a semantic object may sometimes stretch over the sentence boundary and take several utterances to emerge. A spoken dialogue system must therefore be able to deal with semantic objects at a wider range, from subword all the way up to the discourse level. For multimedia systems, the recognition of a semantic object is even more challenging since it can reside over signals from various media.

To cope with these challenges, we employ a representation, called *semantic classes*, to denote the objects and describe the relations that hold among them. Our approach follows the well-established semantic frame method first proposed by [4]. We define the semantics in terms of the dialogue events that may be consequently invoked, i.e., we embrace the spirits of the so-called "procedural semantics" approach [5] in this work. However, we further abstract the notion of semantic frames by typecasting the semantic classes, and use the types of semantic classes as the basic units for constructing syntactic structures and understanding rules. Moreover, the semantic classes are nested recursively to denote semantic objects ranging from the subword to the discourse level, thereby unifying the understanding architecture.

2.1. Type Abstraction

Semantic classes are designed to separate the essential and form-independent attributes of a semantic object from its physical realizations. The attributes of a semantic class can, in turn, be semantic classes themselves. The concept behind semantic classes is identical to the mechanism known as type abstraction commonly employed in software design using a strongly typed programming language. From an understanding point of view, a semantic class is an abstraction of the collection of semantic objects that have the same attributes and usually can be expressed, and hence be understood, in the similar manners.

Another argument for type abstraction is that the multitude of semantic objects is usually a result of the numerous ways and perspectives that can be used to describe a physical entity. Quite often in an understanding system, it is more important to correctly identify the entity of interest than to capture the mechanism that describes it. For instance, one may refer to a person by his name, job function, relations to others, or, in a multimodal environment, by pointing to his photo on a display. All these references lead to semantic objects that are apparently distinct yet should be associated with the same physical entity. Accordingly, it is often useful to segregate the conceptual manifestation and its realizations into different layers of abstraction so that the semantic objects can be better organized and managed.

2.2. Property Inheritance

Introducing inheritance into the semantic class hierarchy further augments the multi-layer abstraction mentioned above. Class A is said to inherit or be derived from class B if class A possesses all the attributes of class B. In this case, class A and B are called the derived class and the base class, respectively. Inheritance is a mechanism to propagate knowledge and properties through the structural relationships of semantic classes. It is crucial for many types of intelligent behavior such as deducing presumed facts from general knowledge and assuming default values in lieu of explicit and specific facts.

Perhaps the strongest motivation to employ inheritance is to facilitate the multi-layer abstraction mentioned above. Very often, a base class is constructed with the general properties of a type of semantic objects, and a collection of more specific classes are derived from the base class to support the various embodiments of the underlying type of the semantic objects. For example, a semantic class architecture for the reference to a person can have the methods (e.g., by name, job function) and the media (e.g. speech, handwriting) of reference as the first layer of derived classes. One can then cross match the viable means (e.g. by name via speech, by name via handwriting) and develop the second layer of derived classes for use in the real applications.

2.3. Functionality Encapsulation

The goal of abstraction is to reduce the complexity in describing the world, in this case, the semantic objects and their relations. One can inspect the quality of abstraction by examining the extent to which the constructs, i.e., semantic classes, are self-contained, and how proliferating they have to become in order to account for novel scenarios. Studies in data structure and software engineering propose the notion of encapsulation, which suggest that individual attributes have local rather than global impacts. This principle also serves as a guideline in designing the semantic class.

Semantic class encapsulation can be elaborated in two aspects: syntactic and semantic. The syntactic encapsulation refers to the constraint that each attribute in a semantic class can only

have relations to others from the same class. The collection for these relations is called the *semantic grammar,* which specifies how a semantic object of this type can be identified in a signal. The semantic encapsulation, on the other hand, dictates the actions and the discourse context under which they may be taken by a semantic class. This is discussed further in Sec. 3.

2.4. Dialogue Polymorphism

Dialogue polymorphism refers to the fact that a given semantic grammar may elicit, in an automatic manner, a variety of actions that are dependent upon for various contexts. The versatility in system responses is a key measure of success for an intelligent dialogue system. In our model, dialogue polymorphism is achieved naturally by combining the two aforementioned mechanisms: encapsulation and inheritance. In dealing with a person, for example, the grammar may simply point to the base class mentioned in Sec. 2.2. As the user's response is collected, the signal is parsed with the semantic grammars from all the derived classes, and the best match is chosen to instantiate a semantic object. The system can accordingly carry out the encapsulated actions that are attached to the chosen semantic class.

3. DIALOGUE SYSTEM

In a plan-based dialogue model, the dialogue actions are regarded as a subset of the activities involved in the process of semantic evaluation. By taking the notion of procedural semantics, we equate semantics with actions that follow consequently, and encapsulate them in the semantic class. The dialogue management system in the proposed model (Fig. 1) is responsible for following through these actions. In this section, we elaborate two aspects to the issue, namely, the mechanism and the order in which these actions are executed.

3.1. Dialogue Events

It is important to note that the proposed dialogue model is intended to be domain and application independent. In other words, our focus in this work is a generic dialogue rendering architecture. The competency of the system resides solely in the specification of semantic classes, but that should not prohibit a design of an intelligent system. The key issue is that, because the dialogue system can make no assumption on what kind of actions it may encounter, general mechanisms must be provided for executing actions.

We observe that general purpose GUI rendering also faces the similar problem, and a solution, i.e., the event based approach, has been widely used with much success. We believe that the event model may also be adapted to realizing intelligent dialogue systems. For our purposes, we define the following dialogue events, through the capture of which the domain dependent actions can be invoked.

Active Instantiation

In a mix initiative dialogue system, a semantic object can be instantiated either because new information voluntarily offered by the user is received, or because it is the natural action deduced by the dialogue system. From the system's point of view, these two mechanisms are called passive and active instantiation, respectively.

Passive instantiations are system internal actions in the sense that they are fully specified one the semantic class architecture is given. Active instantiations, on the other hand, usually involve synthesis of a series of actions to solicit user's response and therefore are application dependent.

Semantic Evaluation

This event is posted by the system whenever it requires domain specific knowledge to proceed in the course of evaluation. A need for database inquiry, for instance, is a common cause that prompts the dialogue manager to issue this event.

The semantic evaluation requires the most sophisticated event handler because it must send back to the dialogue system messages that usually change the flow of the dialogue, or trigger other events (dialogue repair or semantic binding) to be posted. Messages that can be returned are evaluation succeeded, evaluation failed, invalid information, and value to be determined. Sec. 3.2 further discusses how these affect the dialogue flow generation.

Dialogue Repair

A dialogue may be diverted away from the ideal flow due to various reasons (e.g. mis-recognition, out of domain reference, conflicting information), many of which require domain and application specific knowledge to guide the dialogue back to the desired course. This process is called dialogue repair. An appropriate repair strategy is often critical to the usability of a dialogue system, but few guidelines are domain and application independent. Accordingly, our system only detects the need for dialogue repair, and leaves the realization of the repair strategy in the corresponding event handler.

Semantic Binding

Semantic binding refers to the step that links the semantic object, after being instantiated through the evaluation process, to the physical entity in the application. Because an entity can be identified by partial information (e.g., last name of a person), binding is necessary for the system to grasp the whole attributes of the objects the dialogue is concerned with. Semantic binding is also critical for intelligent behaviors such as setting the discourse context for reference resolution. In this model, the binding event handler is also an ideal place to include the actions, such as greeting or confirmation, to enhance the flow and user experience.

3.2. Flow Inference

One common belief in intelligent dialogue studies is that the dialogue flow can be generated from the discourse context and the relations among the semantic objects by means of logical inference. Accordingly, to facilitate automatic flow generation, one has to define not only what attributes should be grouped together to form a semantic class, but also how these attributes should interact with each other in a dialogue.

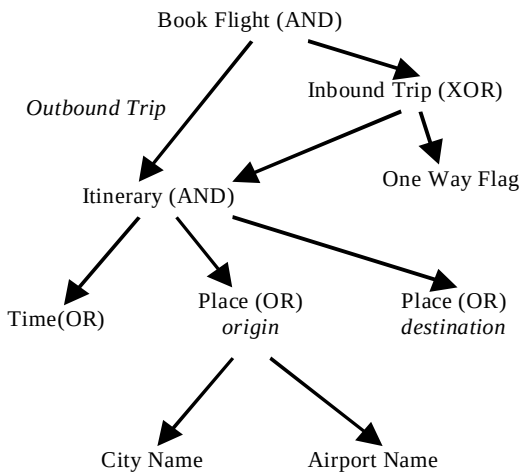

Figure 2: A portion of the semantic tree hierarchy for the Airline Travel Information System application.

To this end, we introduce the concept of "logical container" as a dialogue property to be encapsulated in a semantic class. Three types of logical containers are defined. A semantic class is an AND type container if all its attributes must be evaluated successfully. If this requirement is not met, the evaluation of the AND type semantic object is considered failed, which will prompt the system to post a dialogue repair event. An OR type container requires at least one attribute to be successfully evaluated. Similarly, for an exclusive or (XOR) type container, one and only one. All these three containers operate on the short-circuit Boolean logic.

Fig. 2 illustrates a partial semantic class hierarchy for a simple application similar to the Airline Travel Information System. The dialogue goal, to gather the information for booking a flight, corresponds to the highest level semantic class. Evaluating this semantic class drives the dialogue system to traverse down the semantic class structure, eventually fulfilling all the steps necessary to achieve the dialogue goal. This is achieved by recursively evaluating the attributes, instantiating semantic objects actively if necessary. The logical relation of each semantic class determines the rules of instantiation and dialogue repair. For instance, if the user specifies the trip to be one way only, the evaluation of the "One Way Flag" semantic class will succeed. As the "Inbound Trip" semantic class being an XOR container, the dialogue system will bypass the evaluation of the "Itinerary" attribute in the "Inbound Trip" semantic class.

The "Itinerary" semantic class encapsulates the basic elements to specify a one way trip: the time of travel, the origin and the destination of the trip. Since it is designated as an AND type container, the dialogue manager will try to acquire any missing information by actively instantiating the corresponding semantic classes it contains. The active instantiation event handlers for these classes solicit information from the user by implementing certain prompting strategy.

The "Place" semantic class, which is used to denote both the origin and the destination, is designed to pinpoint the exact location for air travel. The user may specify the location by either the city name or the airport name. One convention here is that if multiple airports serve the city, the evaluation of the "City Name" semantic class will fail. The method effectively instructs the dialogue system whether an ambiguity due to multiple airports has occurred, and an airport semantic object should be actively instantiated to resolve the ambiguity.

4. CURRENT PROGRESS

At this time, we have implemented the core rendering engine to communicate in speech, text inputs from keyboard, and text outputs on the screen. For speech modality, an automatic recognizer and a text to speech synthesizer are used to convert speech into text and vice versa. A dynamic programming algorithm is implemented to parse the text inputs into the semantic object structure. Heuristic based scoring methods are currently used. In the near future, we intend to eliminate the need to convert the signal into a textual representation by implementing semantic classes appropriate for various media.

5. SUMMARY

In this article, we describe dialogue system architecture aimed at multimodal human machine interactions. The system consists of a signal understanding and a dialogue management module, both of which are considered to be complementary part of the semantic evaluation process. The understanding system is responsible for detecting the semantic objects embedded in the signal, based on which the dialogue system then attempts to extract the semantics at the discourse level. Since the system is intended to be generic, protocols must be defined for accessing application dependent knowledge. The event model is adopted as the mechanism for this purpose. We define the types of events that are required, and demonstrate their role in the system with an example.

6. ACKNOWLEDGEMENT

Many core features of the semantic classes are originated from Bruno Alabiso at Microsoft Research.

7. REFERENCES

1. Sadek, M.D., Bretier, P., and Panaget, F., "ARTIMIS: Natural Dialogue Meets Rational Agency," *Proc. IJCAI-97*, Japan, 1997.

2. Allen, J.F., *Natural Language Understanding*, 2nd Ed., Benjamin-Cummings, Redwood City, CA, 1995.

3. Cohen, P.R., Morgan, J., and Pollack, M.E., *Intentions in Communications*, MIT Press, Cambridge, MA, 1990.

4. Minsky, M., "A Framework for Representing Knowledge," in *The Psychology of Computer Vision*, edited by P. H. Winston, McGraw-Hill, New York, NY, 1975.

5. Winston, P.H., Artificial Intelligence, 3rd Ed., Addison-Wesley, 1992.