A PLAN-BASED DIALOG SYSTEM WITH PROBABILISTIC INFERENCES

Kuansan Wang

Speech Technology Group, Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA

ABSTRACT

In this paper, we present a dialog system that extends the plan-based approach with two features. Instead of Boolean inference, we include into the system the probabilistic measures from the front-end speech and language processes. As a result, rules can be activated and facts gathered based on statistical confidence measures. We also introduce the notion of entity types to classify the rules and facts. The entity types, derived from the schema of the knowledge base, assist the semantic evaluation process by indicating which rules and facts are interoperable. The semantic evaluation and dialog planning can therefore be better insulated among tasks, and be encapsulated into reusable components. To assess the feasibility and discover the areas for improvements for this framework, we launched the project DR. WHO, in which we strive to develop a speech interface for a personal information management application. We describe the current progress in the discourse area in this paper.

1. INTRODUCTION

It is widely acknowledged that, in order to create scalable dialog applications, one must prepare the computer with enough intelligence to automatically determine the proper course of actions. Plan-based systems, such as [1-3], are designed with this idea in mind. The mathematical foundation of the plan-based approach is logical inference. The behaviors of the system and the knowledge of the domain are programmed as a set of logical rules and axioms. The system interacts with the user to gather facts, which consequently trigger rules and generate more facts as the interaction progresses. One can lump a set of axioms and rules into self-contained modules (idioms) that serve as reusable building blocks for new applications. The logical containers [4], for example, are a possible implementation.

Although powerful, the plan-based approach has the following difficulties that prevent it from wide adoption. First, authoring the logical rules and axioms often requires human experts that are well versed in both the system design and the application domain. The performance of the system correlates highly to the quality of the rules, and the skills of writing high quality rules are hard to acquire. Secondly, the plan-based infrastructure resting upon Boolean logic often implies making "hard" decision. In many cases, a softer, or "fuzzy" logic is more appropriate. This is even more applicable to a spoken dialog system, in which the speech recognition results are usually manifest as a list of alternative hypotheses with various degrees of likelihood. Finally, the proper system behaviors often require iterative refinements based on trial results or field data. These data can be either too overwhelming for human experts to digest, or sometimes outright unavailable due to technical or privacy concerns. As a result, it is highly desirable for a plan-based system to be equipped with the data-driven capabilities that are suitable for on-line adaptation. The idea of data-driven dialog system is not new. For instance, statistical methods have been applied to acquire dialog strategies [5], semantic parsing [6], or both [7]. We would like to embrace the notion of fuzzy decision implied by the statistical methods and extend it to the whole system that seamlessly integrates the strengths of a plan-based approach.

The extent to which our approach applies dwells within the boundary where an objective and analytical performance measure makes sense. A dialog system, however, inevitably include areas where the ultimate performance measure is a subjective one. To that end, we further distinguish the notions of response planning and rendering. In a planned-based system, response planning is a natural outgrow of the semantic evaluation process. It is the step where the system's intent is computed. The outcome of the planning process is a message the system would like to convey to the user. Ideally, the message should point to a proper course of action that is independent of the physical environment in which the user interacts with the system. The response generation, in contrast, is the process in which the message is physically presented to the user. This is the stage mostly susceptive to application specific and user interface considerations. To handle a message requesting the user to select a sizeable list of alternatives, for example, a system with a suitable visual display might choose to present the whole list, while a speech only system might require a more clever strategy and resolve the ambiguity in more dialog turns. For the scope of this paper, we will focus on the system components preceding the rendering stage. An implementation of the rendering stage using the framework described in this paper is described in [12].

2. SYSTEM OVERVIEW

The key idea in this work is to augment the decision process in the plan-based system from using solely the Boolean logic to probabilistic measures. Essentially, the rules being triggered no longer just depend on whether all the predicates are satisfied, but also the *likelihood* of the predicates. From this perspective, one may also argue a probabilistic plan-based dialog problem is a step-wise pattern recognition problem that can be stated as follows. Given an input x (natural language text or speech), the objective of the system is to arrive at an action A so that the cost of choosing A is minimized. With a proper categorization, one can assuming uniform cost, which leads to the optimal solution to be the maximum *a posteriori* (MAP) decision

$$\begin{aligned} A_{opt} &= \arg\max_A P(A \mid x, S_{n-1}) \\ &\approx \arg\max P(A \mid S_n) \sum_F P(S_n \mid F, S_{n-1}) P(F \mid x, S_{n-1}) \end{aligned} \tag{1}$$

where F denotes a collection of semantic objects (Sec. 2.2) of x and S_n, the discourse semantics for the n^{th} dialog turn. Based on this formulation, a dialog system is basically composed of three components: a semantic parser that converts x into a collection of semantic objects, a discourse manager that derives new dialog context based on the per-turn semantic parse and the previous context, and a response manager that picks the most suitable action from all the possibilities and renders it to the physical device. The probabilistic measures governing the operations of response, discourse, and the parse are called the behavior model $P(A|S_n)$, the semantic model $P(S_n|F, S_{n-1})$, and the language model $P(F| x, S_{n-1})$, respectively.

The equation above seems to suggest that a plan-based system is merely an embodiment of a statistical state machine for which the discourse semantics are regarded as states. The difference, however, is that the "states" for the plan-based system are generated dynamically and not limited to a pre-determined finite set. This capability of handling unbounded number of states is a key strength of plan-based systems in terms of scalability and flexibility.

2.1 Knowledge and Semantic Classes

Since it is crucial to utilize domain knowledge at every possible step during the semantic evaluation process, we follow the principles outlined in [4] to convey domain knowledge in the system. We assume that the domain knowledge conforms to a relational or objected oriented database, of which schema is clearly defined. We use the term *entity* to refer to a data item in the domain (a row in a database table), or a function (command or query) that can be fulfilled in the domain. A column in the database table is called an entity attribute, and each database table is given an entity type. If the domain is characterized by an objected oriented database, the inheritance relationships among the entity types follow those of the database.

Through a small subset of its attributes, an entity can be realized linguistically in many ways. We call each one of them a *semantic class*. For example, a person can be referred to with his full name (*"Xuedong Huang"*), a pronoun anaphora (*"him"*), or through relationships to others (*"Kuansan's manager"*). In this case, one can derive three semantic classes for the entity type "person." Note that the semantic class can be recursive, as demonstrated in the last example that a "person" semantic class contains an attribute of "person" type. Since the entities can be nested, i.e., a database column can in turn refer to another table, an attribute in the semantic class can also be an entity type. The main motivation of having multiple semantic classes for each entity type is to better encapsulate the language, semantic, and behavior models based on the domain knowledge. While the entity relationships capture the domain knowledge, the semantic class hierarchy represents how knowledge can be expressed in the semantics of a language.

2.2 Semantic Parser

An instantiation of a semantic class is called a semantic object. Due to the nested nature of semantic classes, a semantic object F in Eq. (1) can itself be a tree of semantic objects. The attributes of a semantic object are either domain entities or semantic objects. One may view the semantic object attributes as the predicates in the plan-based model. A user's utterance may also consist of disjoint fragments that only make sense at the discourse level. For instance, in the context of setting up a meeting, the user utterance "*Kuansan Wang at a quarter to two*" can be parsed into two objects: a person and the meeting time.

Parsing is essentially a dynamic programming problem. The language model defines how attributes are phrased. We have been experimenting with two types of language models: context free grammar with robust parsing [8], and, more recently, the unified grammar [9]. The unified grammar is an N-gram of terminal and non-terminal tokens. Terminals are words in the lexicon. Non-terminals are the unified grammars associated with all the attributes contained by the semantic class. If an attribute is an entity type, its unified grammar includes the union of all the semantic classes of this type.

At the end of each system's turn, the discourse manager provides the parser with "dialog focus," a list of semantic objects and classes that, based on the current dialog context, are most likely to be instantiated in the following turn. Effectively, dialog focus creates a dialog state dependent language model. The parser uses the dialog focus to bias the grammar in forming and scoring the parse trees.

2.3 Discourse Semantics Evaluation

The discourse manager maintains a stack of discourse trees and an entity memory that are operated under a two-step algorithm. The discourse tree has the same form as a parse tree. The discourse is designed to assume a tree structure so that the representation remains the same whether the information is obtained through several dialog turns or a single one.

During the expansion phase, the discourse manager affixes the semantic objects as branches to the discourse tree. Ambiguity arises when a parse can be attached to more than one node on the discourse tree. Ambiguity is resolved either by using the parse scores or by triggering a dialog event. As indicated in Eq. (1), we currently use Viterbi approximation for the discourse semantics, i.e., only the discourse tree with the highest score, rather than a weighted sum of all possible interpretations, determines the system's response. The goal here, however, is to collapse the discourse tree by resolving the semantic objects into the domain entities. Following the expansion is the evaluation phase, in which the discourse manager taps into the knowledge base with the semantic object attributes. The semantic objects to entity conversion proceeds from the leaves up towards the root of the discourse tree. The process ends when the root node is converted, which indicates the dialog goal has been achieved. Whenever a conversion occurs, the consequent entity is added to the entity memory. Here we adopt the chunking memory model [10] in implementing the entity memory. The entity memory consists of turn and discourse memories. Either type of memory consists of a number of priority queues that are delineated by entity types. An entity can only be remembered into the queue of compatible types (e.g., through inheritance). When referred to, the memory item will increase its priority in the queue. The *turn memory* is a cache for holding entities in each turn. There are two types of turn memories. The *explicit* memory holds the entities that are resolved directly from semantic objects. In contrast, the *implicit* memory is for entities that are deduced from anaphora, deixis, or ellipsis. In accessing the memory, the explicit turn memory takes precedent over the discourse memory, which in turn has a higher priority than the implicit. At the end of the system's turn, all the turn memory items are moved and sorted into the discourse memory. The distinctions between the three kinds of memories and the rules to operate them are designed as a simple mechanism for most common but not all the possible scenarios. It is worth noting that the design has a bias towards direct and backward reference. For example, in the expression "*Forward this mail to John, his manager, and his assistant*", the second "*his*" will be evaluated as referring to John, not John's manager. The implicit memory, however, provides a back off for expression like "*Send email to John, his manager, and her assistant*" in which the pronoun "her" should be taken as indicating John's manager is a female and resolved accordingly. However, since we only store the entities and not the semantic objects into the memory, the mechanism is not suitable for forward or default references, as in the examples like "*Since his promotion last May, John has been working very hard*" or "*It being so nice, John moved the meeting outside*." Fortunately, these natural language phenomena are rare in a spoken dialog environment.

The evaluation phase begins at converting the semantic objects whose attributes are all filled. Note that in our system, anaphora and deixis are handled with specialized semantic classes. In addition to memory manipulation, semantic classes can also be decorated with behaviors on inference. An "auto" semantic class, in particular, is designed to automatically infer information not supplied by the user. For instance, a meeting entity, once resolved, can be later referred to in expressions such as "*the morning meeting*", "*meeting with Kuansan*", or "*the meeting in building 113*" that, on the first encountered, are nevertheless ambiguous. In our system, the strategy to automatically resolve partially specified entities is as follows. During the evaluation stage, a partially filled, auto semantic object is first compared with the entities in the memory based on the type compatibility. If a candidate is found, the discourse manager then computes a goodness of fit score by consulting the knowledge base and considering the position of the entity in the memory list. The semantic object is converted immediately to the entity from the memory if the score exceeds the threshold. In the process, all the actions implied by the entities are carried out following the order the corresponding semantic objects are converted.

Often in the design process, we find it desirable to segregate a dialog into several self-contained sessions, so that each of which can employ specialized language, semantic, and even behavior models to further improve the system performance. These sessions are sub-goals of the dialog, which usually manifest themselves as "trunk" nodes on the discourse tree. We implement a tree stack in which each trunk node is treated as the root for a discourse tree. The stack is managed in a first-in last-out fashion as currently no digression is allowed from one sub-dialog to another. So far, the no-digression rule is considered a reasonable trade-off for dynamic model swapping.

A system response is needed whenever the dialog goal has been reached or some semantic objects cannot be successfully converted. The task for the discourse manager here is to send to the response manager a universal message that suggests the proper course of action. At the end of the above evaluation process, the discourse manager tabulates the remaining semantic objects and examines the causes that prevent them from being converted. We score these causes with domain knowledge and their proximity to the current focus. Most often, the response is either a confirmation or a negotiation. A confirmation occurs when a semantic object has obtained all its attributes, but the score is too low to accept it outright as an entity. In this case, the discourse manager simply presents the score to the response manager, which then decides whether an explicit or implicit confirmation is appropriate. A negotiation response can arise whether a semantic object is fully filled or not. For under-specified semantic objects, possible actions include simply to pursue the unfilled attributes in a predefined order, or to gather the entities in the knowledge base sorted by various keys. For cases of ill specification, an entity that matches the semantic object attributes does not exist. The discourse manager can simply report such fact, or suggest removal or replacement of certain attributes, depending on how much domain knowledge to be included in the planning process.

3. DR. WHO'S IMPLEMENTATION

We implement a dialog system for project DR. WHO [11]. Our first effort aims at providing the four mobile applications, i.e., messaging, scheduling, directory, and reminders, with a spoken language interface on a palm-size computer. The system is implemented to render the responses on the display, although the design also includes synthesized speech rendering for eyes-busy or other displayless usages [12]. When used with a pointing device (e.g., a stylus or a roller), the user can point to a specific area of the screen before issuing speech commands. This user-assisted focus forming, known as the tap-and-talk feature, allows the system to dynamically swap in semantic grammar that has tighter coverage.

We use the object models of Microsoft Outlook as the foundation for the knowledge base. Entity types related to the logical objects, such as messages and appointments, are directly obtained from the object models. We also derive the functional entities, such as reading email and setting up a meeting, from the capabilities exposed by these object models. For the directory task, we also develop a middleware using the Microsoft Access object model to connect to the human resource database on the corporate intranet. The entity type "person," based on the Access database, contains an email address attribute that serves as the cross-reference key for the people objects (e.g., attendees of a meeting, email recipients) in the Outlook database. The directory task provides a good glimpse of the system since it is the most compact and originates widely reused components. The task is modeled by a single semantic class with two attributes that can be described in XML as

```
<class type="Task" name="Directory" inference="auto">
```

```
    <slot type="Person"/>
    <slot type="DirectoryItems"/>
    <expert ref="./directory.dll"/>
  </class>
```

The semantic class has an entity type "Task", indicating it can be a dialog goal for the DR. WHO application. It is declared as an auto-inference semantic class, so the discourse manager will try to resolve missing attributes from the memory. Its first attribute, the target person, is declared by the entity type, which can match all the semantic objects modeling that type (Sec. 2.1). The second attribute is a list of personal properties (e.g., phone number, office location, manager). As a common technique, we use a right-recursive semantic class to form a list:

```
<class type="DirectoryItems" memory="explicit">
  <slot type="DirectoryItem"/>
  <slot type="DirectoryItems"/>
</class>
```

Finally, the semantic class is supplied with the domain expert that interacts with the database and plans the responses. A sample text-based dialog goes like this:

U: What is the phone number for Dan?
S: Many Dan's are in the database. Please say the full name.
U: Dan Venolia.
S: The number is 4257032891.
U: Where is his office?
S: The office is in building 31, room 1362.
U: How about Kuansan's?
S: The office is in building 31, room 1363.
U: Who is Dan's manager?
S: Xuedong Huang.
U: What are his office and phone numbers?
S: The office is in building 31, room 1332.
S: The phone number is 4259363966.

Note that the system resolves the ambiguous name "Dan" only once, even though the reference to "Dan" is interjected. The second reference is automatically resolved since the semantic class "Directory" is declared as "auto" and an entity that matches the partial reference can be found in the memory. In contrast, no clarification is needed for "Kuansan" since there is only one person in the database that has "Kuansan" as either the first name or last name. As shown above, we declare the semantic object of directory items to be sent to the explicit memory once it is resolved. Accordingly, when the user only mentions a person in a query, the items in the memory are assumed by default, as in the response to the utterance "*How about Kuansan's?*"

4. SUMMARY

In this paper, we propose a framework that regards the plan-based dialog approach as a cascade of statistical pattern recognizers. At the semantic level, the patterns to be recognized are what we refer to as the entities that, basically, are speech acts or references. We define the linguistic constructs of the entities as semantic objects. Consequently, at the language understanding front-end, the patterns to be recognized are the semantic objects. Finally, once the discourse semantics is reached, the patterns to be recognized in the response generation stage are the desired course of actions. We argue the predicates in a plan-based system can be implemented via semantic objects.

We believe that introducing the statistical measures into the plan-based approaches will further strengthen the robustness of the system without comprising its scalability. We also introduce the notion of semantic objects that are tightly related to the schema describing the domain knowledge. By "objectifying" the rules governing the semantic evaluation process, we hope to make the components in the system more reusable, and hence more extensible. We implement a prototype based on these ideas, and so far, the results seem favorable and encouraging.

Several aspects in our current prototype require further investigations. As indicated in Sec. 2, we employ Viterbi approximation in inferring the discourse semantics. However, as is reported in [7], it is worthwhile to maintain alternative semantics for response planning. In addition, one key strength for the plan-based approach is it provides a natural pathway towards multimodal integration. We have not fully explored this area yet, but are looking forward to it. Finally, the behavior and semantic models in the prototype currently use hard-coded heuristic scores to approximate the probabilities. We still have to establish how much gain we can derive from data-driven, well-trained models.

5. ACKNOWLEDGEMENT

The interfaces to Outlook are implemented with the assistance of Ricky Loynd.

6. REFERENCES

[1] Allen J.F., Ferguson G., Miller B., and Ringger E., "Trains as an embodied natural language system," *Proc. AAAI-95 Symposium on Embodied Language and Action*, 1995.

[2] Sadek M.D., Ferrieux A., Cozannet A., Bretier P., Panaget F., and Simonin J., "Effective human-computer cooperative spoken dialogue: The AGS demonstrator," *Proc. ICSLP-96*, 1996.

[3] Cohen P.R. and Levesque H.J., "Communicative actions for artificial agents," *Proc. ICMAS-95*, San Francisco, 1995.

[4] Wang K., "An event based dialog system," *Proc. ICSLP-98*, 1998.

[5] Levin E., Pieraccini R., and Eckert W., "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Trans. on Speech and Audio Processing, pp.11-23*, January 2000.

[6] Magerman D. M., "Statistical decision-tree models for parsing," *Proc. ACL-95*, 1995.

[7] Souvignier B., Kellner A., Rueber B., Schramm H., Seide H., "The thoughtful elephant: strategies for spoken dialog systems," *IEEE Trans. on Speech and Audio Processing, pp.24-36*, January 2000.

[8] Wang Y.-Y., "A robust parser for spoken language understanding," *Proc. EuroSpeech-99*, 1999.

[9] Wang Y.-Y., Mahajan M., Huang X., "A unified context-free grammar and N-gram language model for spoken language processing," *Proc. ICASSP-2000*, 2000.

[10] Baddeley A., *Working Memory*, Oxford University Press, 1986.

[11] http://research.microsoft.com/stg/drwho.htm

[12] Wang K., "Implementation of a multimodal dialog system using extensible markup languages," *Proc. ICSLP-2000*, 2000