Semantic Object Synchronous Understanding in SALT for Highly Interactive User Interface

Kuansan Wang

Speech Technology Group, Microsoft Research
One Microsoft Way, Redmond, WA 98052 USA
http://research.microsoft.com/stg

Abstract

SALT is an industrial standard that enables speech input/output for Web applications. Although the core design is to make simple tasks easy, SALT gives the designers ample fine-grained controls to create advanced user interface. The paper exploits a speech input mode in which SALT would dynamically report partial semantic parses while audio capturing is still ongoing. The semantic parses can be evaluated and the outcome reported immediately back to the user. The potential impact for the dialog systems is that tasks conventionally performed in a system turn can now be carried out in the midst of a user turn, thereby presenting a significant departure from the conventional turn-taking. To assess the efficacy of such highly interactive interface, more user studies are undoubtedly needed. This paper demonstrates how SALT can be employed to facilitate such studies.

1. Introduction

The Speech Application Language Tags (SALT) [1] was introduced to enable speech as a viable input/output modality for computer user interactions. The design goal for SALT is to make common speech tasks simple to program, yet allow advanced capabilities straightforward to be realized. SALT was designed with many application scenarios in mind. An important area, for example, is the telephone-based speech-only applications where the computer interacts with the user exclusively through spoken dialogs (e.g. [2][3][4]). Because telephone users enjoy full duplex conversations among one another, it is natural for human to expect the spoken dialog systems to behave the same way. The computer is expected, for example, to be able to manage the dialog turn-taking by automatically end-pointing a user turn. As a result, the SALT speech input and output objects, called *listen* and *prompt*, respectively, both have a mode designed to incorporate technologies that can detect the start and the end of a user turn. Except for the simple cases, however, automatically end-pointing dialog turns is still an actively researched topic [5][6][7][8]. Many speech applications today employ an interface design that requires the user to signal the start of a user turn by, for instance, pushing a button. Examples include wearable computers (e.g. [9][10]), speech enabled portable or multimodal devices (e.g. [11] [12][13]), and other eyes free applications (e.g. automobile). SALT objects are equipped with rich controls that allow programmers to produce rich UI effects for these environments. But despite the variations in the interface designs, these application areas all seem to base on a framework that clearly defines the boundaries of the user and the system turns. Many dialog evaluation techniques are established on this notion as well [14][15].

In one SALT use case, however, we observe that the notion of dialog turns may be ill defined. Specifically, we zoom in on the so-called *multiple* mode of SALT recognition in which the active *listen* object can raise events on partial understanding while the underlying audio capturing and speech recognition are still in progress. Because SALT returns semantic objects sufficient for dialog logic execution, this enables the computer to immediately react and report outcomes based on the partial utterance, even before the end of the user turn. In a way, one can view the use case as one in which the back channel communication is augmented to perform tasks normally associated with a system turn, thereby blurring the boundary of a user and a system turn. Most conventional dialog studies, especially those based on human to human dialogs, often view the back channel communications as non-intrusive feedbacks that convey only simple signals such as positive, negative, or neutral acknowledgement [16][17] [18]. However, the feedbacks from the *multiple* mode in SALT can potentially carry more information that, when presented immediately to the user, may influence the ongoing user utterance. This exploitation therefore seems to manifest itself as a departure from the classical turn-taking view of dialog management, and more studies, especially in the area of usability, are undoubtedly needed to assess the viability of this method. One purpose of this paper, however, is to show how SALT can be programmed to perform semantic-object synchronous decoding and report immediate partial understanding so that prototypes for user experiments can be quickly created.

The rest of the paper is organized as follows. In Section 2, we first describe our treatment of the speech understanding problem using the speech recognition framework. The technique we use is an enhanced context free grammar with semantic tagging capability. Semantic tagging allows the grammar creator to normalize the potentially complicated syntactic variations the CFG is designed to cover. Moreover, we demonstrate how the semantic tagging can even be applied to a semantic language model where N-gram is introduced to model pre-terminals that are difficult to be covered well by CFG rules. With these enhancements we show in Section 3 how the SALT *listen* object can be configured to perform speech understanding tasks in a semantic object synchronous manner even though the object is primarily designed for speech recognition purposes.

2. Surface semantic extraction

The SALT *listen* object can be used to perform both speech recognition and understanding tasks. This is possible because the design follows the formulation of [13][19] that treats speech understanding as a pattern recognition problem, just like speech recognition: for speech recognition, the pattern to be found is a string of words and, for understanding, a tree of semantic objects. A traditional speech recognition task instructs the search process with a language model to compose the likely word strings. In the similar fashion, the search engine can be guided to compose the suitable semantic object trees with a semantic model [13]. Like a language model that often implies a lexicon and the rules of composing phrase segments from the lexicon entries, a semantic model implies a dictionary of all semantic objects and the rules of composing them. Many other works, such as the unified language model [20] or a weighted finite state transducer [21], have also demonstrated that recognition and understanding can share the same search algorithm and engine implementation.

Although it is possible to extend N-gram to return a structured search outcome (e.g. [23]), we base our speech understanding framework on the probabilistic context free grammar (PCFG) where the grammar rules for composing semantic objects can be authored without massive tree-bank annotated training data. One method of specifying such rules is to associate each PCFG rule with instructions for the search engine to transform the partial PCFG parse tree into a semantic object tree [24][26]. An example written in Microsoft Speech Application Interface (SAPI) XML format is shown below:

```
<rule name="nyc">
 <list>
   <phrase>new york ?city</phrase>
   <phrase>?the big apple</phrase>
 </list>
 <output>
  <city_location>
   <city>New York</city>
   <state>New York</state>
   <country>USA</country>
  </city_location>
 </output>
</rule>

<rule name="NewMeeting">
 <ruleref min="0"   name="CarrierPhrase"/>
 <ruleref max="inf" name="ApptProperty"/>
 <output>
   <NewMeeting>
    <DateTime>
      <xsl:apply-templates select="//Date"/>
      <xsl:apply-templates select="//Time"/>
      …
    </DateTime>
    <Invitees>
      <xsl:apply-templates select="//Person"/>
    </Invitees>
   </NewMeeting>
 </output>
</rule>
<rule name="ApptProperty"/>
 <list>
    <ruleref name="Date"/>
    <ruleref name="Duration"/>
    <ruleref name="Time"/>
    <ruleref name="Person" max="inf"/>
    <ruleref name="ApptSubject"/>
   .. ..
 </list>
</rule>
```

The grammar segment contains three rules. The first one, a pre-terminal named "`nyc`" lists the possible expressions for New York City. The `<output>` tags in this example enclose the rules for constructing semantic objects. They are invoked when the search path exits the grammar node denoted by the token immediately preceding it. In the case, a semantic object, represented in XML with a `<city_location>` element, is created when a search path exits the "`nyc`" rule. This semantic object is in turn composed of three semantic objects: the city name, state and country name abbreviation. The composition of semantic objects can also be a dynamic process, as demonstrated in the second rule used in MiPad [11] for scheduling a new meeting. Here, a `NewMeeting` semantic object will be produced when the user finishes specifying the meeting properties such as date, time, duration and attendees. The XSLT [25] `apply-templates` command is used to paste other semantic objects as constituents into the `NewMeeting` semantic object. As an example, an utterance "schedule a meeting with Li Deng and Alex Acero on January first for one hour" will result in the following semantic object:

```
<NewMeeting>
  <DateTime>
    <Date>01/01/2003</Date>
    <Duration>3600</Duration>
  </DateTime>
  <Invitees>
    <Person>Li Deng</Person>
    <Person>Alex Acero</Person>
  </Invitees>
</NewMeeting>
```

Generally, improving PCFG coverage is a daunting task. It is therefore desirable that one can use the N-gram to model the *functional phrases*, for example, that do not carry critical semantic information but usually have sizeable variations in the syntactic structure (e.g., "May I…", "Could you show me…", "Please show me…"). We have utilized a technique, called semantic language model [13], to combine PCFG with N-gram. The technique is slightly different from the unified language model work previously published in [20] although both appear to integrate PCFG and N-gram into a single recognition process. The unified language model technique is a natural extension to the conventional class N-gram except it allows CFG fragments, not just a list of words, to be modeled as an individual token in N-gram. The recognizer using this model still produces text string that has to be subsequently parsed. The unified language model thus is designed to incorporate certain linguistic structure to assist text transcription.

The semantic language model, on the other hand, aims at using the decoder to search for the semantic structure, which is usually better captured by PCFG. Therefore, instead of embedding CFG fragments into N-gram, we allow the PCFG to model the pre-terminals with N-gram. In SAPI grammar format, a special pre-terminal is specified with an XML `<dictation>` tag, as in

```
LCFG <dictation max="inf"/> RCFG
```

where LCFG and RCFG denote the left and right context of the embedded N-gram. The PCFG search process treats the `<dictation>` tag as a token and expands into the N-gram as if entering a regular non-terminal. The max attribute on the tag specifies the maximum number of words that can be drawn from the N-gram. Inside this N-gram, the word string probability is computed by constantly considering a backoff to PCFG along side with staying with N-gram, namely

$$P(w_n \mid w_{n-1}, w_{n-2}, ...) = \begin{cases} \lambda P(w_n \mid Ngram, w_{n-1}, w_{n-2}, ...) \\ (1-\lambda) P(w_n \mid RCFG) P(RCFG \mid w_{n-1}, w_{n-2}, ...) \end{cases} \quad (1)$$

where λ is the N-gram weight and $P(RCFG \mid w_{n-1}, ...)$ uses the back-off probability of the N-gram, i.e., we treat w_n as if it is an out of grammar word. Currently, we make the term $P(w_n \mid$

RCFG) to assume only binary value depending on whether the maximum N-gram word string size is reached and the word is in the coverage of the CFG fragment or not. When words drawn from PCFG have a higher probability, paths that really belong to be covered by PCFG have tendency to win over their N-gram counterparts even the maximum N-gram word count is set to infinite. In addition to functional phrases, the embbeded N-gram can also be used to model semantic object with a "dictation" like property. For example, the meeting subject is model in our task as

```
<rule name="ApptSubject">
 <p>  <dictation max="inf"/> </p>
 <output>
   <ApptSubject>
     <xsl:apply-templates select="."/>
   </ApptSubject>
 </output>
</rule>
```

3. Semantic Object Synchronous Decoding in SALT

In contrast to other speech recognition or understanding modes that only process the user utterances with end pointing, the *multiple* mode in the SALT *listen* object further provides a means for the search engines to expose more interactive capabilities to the users by allowing them to report immediately whenever a salient linguistic landmark is reached. Search algorithms based on time synchronous decoding [22] can be employed for this mode in a straightforward manner. For speech recognition, the linguistic landmarks usually correspond to word or phrase boundaries. A SALT *multiple* mode recognition was designed to display the word string as soon as they are recognized, a user interface effect commonly seen in the commercial dictation software. When extended for speech understanding, however, the *multiple* mode can treat the instantiations of semantic objects as linguistic landmarks and report them immediately. This creates an appearance as if SALT is performing a semantic object synchronous understanding.

The apparent interactivity may have considerable amount of impacts on human computer interactions. Take for example the MiPad experiments where we employ a user interface technique called tap-and-talk [11] that requires the user to point and hold the stylus in an input field while speaking to the device. Although the tap-and-talk design visualizes the back channel communication by displaying the volume and a progress bar of the underlying spoken language process, those feedbacks provide only primitive clues to the quality of the spoken language processing in terms of speed and accuracy. This can be potentially more problematic for longer sentences in which errors can propagate to a wider scope that eventually requires more efforts in verifying and correcting the recognition and understanding outcomes. Since the usability studies [11] seem to indicate that long sentences are a key differentiating factor that demonstrates the utility of speech as more than a keyboard enhancement or alternative, a satisfying UI experience is absolutely necessary to the success of using speech as a viable modality. A rule of thumb in UI design is to give users more timely and informative feedbacks to promote the perception that the computer and the user are closely collaborative partners in achieving a common goal. One way to include in the back channel communications more indications about the quality of the underlying spoken language processing is to utilize the timely nature of the semantic object synchronous understanding and report the partial outcomes of the spoken language processing as soon as they are available.

We recreate some of the MiPad UI screens with SALT-enabled HTML pages that can be displayed in a Web browser. MiPad was designed so that the users have the ultimate choice for interaction modalities appropriate for the task at hand. When the speech modality is not used, all MiPad interactions are just like a regular graphical user interface realized in HTML. A new meeting request page, for example, utilizes the HTML `input` object to acquire necessary information, such as date, time, location, subject, and meeting attendees, etc., for creating a new meeting:

```
<input id="subject" …> <br>
<input id="date" …> <br>
<input id="start_time" …> <br>
<input id="end_time" …> <br>
<input id="duration" …> <br>
<input id="attendees" …> <br>
```

MiPad's typical tap-and-talk interaction can be implemented with specialized recognition objects, one for each field, e.g.

```
<listen id="rec_subject" …>
  <grammar src="subject.grm"/>
  <bind targetElement="subject"
     value="//ApptSubject"/>
</listen>
```

By default, a SALT *listen* object completes the recognition by automatically detecting the end of a user utterance. When a recognition event occurs, the SALT *bind* directive is triggered to extract relevant portions of the recognition result and assign them to the proper HTML fields. With the exception of time fields, the assignment target is often a single field in the tap-and-talk interaction model. We note that, in order to account for user initiative actions (e.g., speak ahead), it is sometimes necessary to include grammars for multiple field in MiPad. Extending it further, we include all the recognition grammars into a single listen object, and assign their outcomes immediately using the following SALT program:

```
<listen mode="multiple" …>
  <grammar src="subject.xml"/>
  <grammar src="date.xml"/>
  <grammar src="time_duration.xml"/>
  <grammar src="attendees.xml"/>
  <bind targetElement="subject"
     value="//ApptSubject"/>
  <bind targetElement="date"
     value="//DateTime"/>
  <bind targetElement="start_time"
     value="//start_time"
     targetElement="end_time"
     value="//end_time"
     targetElement="duration"
     value="//DateTime/duration"/>
  …
</listen>
```

The multiple grammars compose a parallel search space for the recognition with a null transition looping back to starting point. With the declaration of `mode="multiple"`, the SALT *listen* object raises a recognition event as soon as a grammar reaches the exit state. The event forks a parallel process to invoke the bind directives in sequence while the underlying audio collection and recognition are ongoing, thus creating the effect to the user that relevant fields on the form are being filled while a spoken command is still being uttered. For the eyes-free applications, speech outputs might be desired. In that case, SALT *prompt* objects can be used to give immediate feedbacks. For example, the following SALT *prompt* object can be used to synthesize response based on the dynamic contents in the date field, and the speech synthesis can be triggered with additional SALT *bind* directives as follows:

```
<prompt id="say_date">
  on <value targetElement="date"/>
</prompt>

<listen …>
  …
  <bind targetElement="date"
     value="//date"
     targetElement="say_date"
     targetMethod="Start"/>
     …
</listen>
```

The net effect is the human feels like talking to another party that jots down and repeats what he hears, as in "Schedule a meeting (*new meeting*) at two (*starting at two o'clock PM*) next Tuesday (*on 10/29/02*) for two hours (*duration: two hours*)." Note that SALT allows designers to attach customized recognition event handlers that perform sophisticated computations beyond the simple assignments as with the SALT *bind* directives. In the above example, the date normalization can be accomplished in the semantic grammar which, however, cannot facilitate advanced reference resolution (e.g., "Schedule a meeting with Li Deng and *his manager*"). For such cases, reference resolution algorithms as proposed in [19] may be implemented as script objects accessible to proper event handlers.

4. Summary

This article demonstrates a method to implement semantic object synchronous understanding using SALT. The essential components include a recognizer that utilizes unified language model to produce semantic objects based on frame synchronous decoding, and a SALT listen object capable of reporting partial hypothesis before the recognizer reaches the end of an utterance. As the semantic objects in the partial hypothesis can trigger dialog actions, applying semantic object synchronous understanding to a dialog system seems to blur the conventional definition of a dialog turn. More studies are needed to assess the viability of immediate feedbacks for the dialog designs. We have found SALT provides sufficient capabilities to implement experimental prototypes.

5. Acknowledgements

The technique of embedding N-gram into PCFG described in Section 2 was independently conceived and first implemented into Microsoft SAPI Reference Engine Version 5 by Fil Alleva.

6. References

[1] K. Wang, "SALT: An XML application for Web-based multimodal dialog management", in *Proc. 2nd NLP and XML Workshop*, Taipei, Taiwan, August 2002.

[2] V. Zue *et al.*, "Jupiter: A telephone-based conversational interface for weather information," *IEEE Trans. Speech and Audio Processing*, 8(1), January 2000.

[3] B. Souvignier *et al.*, "The thoughtful elephant: strategies for spoken dialog systems." *IEEE Trans. Speech and Audio Processing*, 8(1), January 2000.

[4] A. Gorin *et al.*, "How may I help you?" *Speech Comm.*, vol. 23, pp. 113-127, 1997.

[5] R. Sato *et al.*, "Learning decision trees to determine turn-taking by spoken dialog systems," in *Proc. ICSLP-2002*, Denver, Co., September 2002.

[6] L. Ferrer *et al.*, "Is the speaker done yet? Faster and more accurate end-of-utterance detection using prosody," in *Proc. ICSLP-2002*, Denver Co., September 2002.

[7] Q. Li *et al.*, "A robust real-time endpoint detector with energy normalization," in *Proc. ICASSP-2001*, vol. 1, Salt Lake City, UT, May 2001.

[8] L. Huang, C. Yang, "A novel approach to robust speech endpoint detection in car environments," in *Proc. ICASSP-2000*, vol. 3, Istanbul, Turkey, June 2000.

[9] N. Sawhney, C. Schmandt, "Nomad radio: speech and audio interaction for contextual messaging in nomadic environments," *ACM Trans. Computer-Human Interaction*, 7(3), pp. 353-383, September 2000.

[10] A. Rudnicky *et al.*, "SpeechWear: a mobile speech system," in *Proc. ICSLP-96*, Philadelphia, PA., October 1996.

[11] X. Huang *et al.*, "MiPad: A next generation PDA prototype," in *Proc. ICSLP-2000*, Beijing, China, October 2000.

[12] S. L. Oviatt, "Multimodal interactive maps: designing for human performance." *Human Computer Interactions*, vol. 12, pp. 93-129, 1997.

[13] K. Wang, "Semantic modeling for dialog systems in a pattern recognition framework," in *Proc. ASRU-2001*, Trento Italy, 2001.

[14] M. A. Walker *et al.*, "DARPA Communicator Evaluation: Progress from 2000 to 2001," in *Proc. ICSLP-2002*, Denver, Co., September 2002.

[15] M. A. Walker, *et al.*, "PARADISE: A framework for evaluating spoken dialog systems," in *Proc. 35th ACL*, 1997.

[16] H. H. Clark, E. F. Schaefer, "Contribution to Discourse," *Cognitive Science*, vol. 13, pp. 259-294, 1989.

[17] L. Bell, J. Gustafson, "Positive and negative user feedback in spoken dialog corpus," in *Proc. ICSLP-2000*, Beijing, China, October 2000.

[18] L. Cerrato, "A comparison between feedback strategies in human-to-human and human-machine communication," in *Proc. ICSLP-2002*, Denver, Co., September 2002.

[19] K. Wang, "A plan based dialog system with probabilistic inferences", in *Proc. ICSLP-2000*, Beijing China, 2000.

[20] Y. Wang *et al.*, "A unified context free grammar and N-gram model for spoken language processing," in *Proc. ICASSP-2000*, Istanbul, Turkey, June 2000.

[21] M. Mohri *et al.*, "Weighted automata in text and speech processing," in *Proc. ECAI-96 Workshop*, Budapest, Hungary, 1996.

[22] H. Ney, S. Ortmanns, "Dynamic programming search for continuous speech recognition," *IEEE Signal Processing Magazine*, pp. 64-83, 1999.

[23] C. Chelba, "A structured language model," in *Proc. EACL-97*, Madrid, Spain, July 1997.

[24] K. Wang, "Natural language enabled Web applications," in *Proc. 1st NLP and XML Workshop*, Tokyo, Japan, November 2001.

[25] XSL Transformations, http://www.w3.org/TR/xslt.

[26] K. Wang, http://www.w3.org/Voice/Group/2000/MS-SemanticTag.html