The Future of Advanced Dialogue Applications

Simona Gandrabur

Nuance Communications
Montreal, Canada

The advanced dialogue work at Nuance is directed at enabling more natural and intelligent user-machine interactions. The interactions that we are most interested in have the following features:

1. Multiple modalities : Input may be solicited or initiated by the user, in the form of spoken or text-based natural language or gestures and may occur simultaneously or sequentially,
2. Multi-turn logic : Information collection, correction, or understanding happens in the context of multiple interactions.
3. Data based operation : System depends on data, word knowledge, user-profile
4. Dynamic call/interaction-flow : The interaction flow adapts dynamically to all of the above features. It would therefore be impractical to specify all possible call flow paths with a finite state machine.

Below is an example of the kind of interactions Nuance is working on within its advanced dialogue/multi-modal projects:
UsrIn-speech>*I want to go from here to there.*
UsrIn-touch>[*click:location1][click:location2]*
AppOut-speech>*This would be the shortest path*
AppOut-screen>[*itinerary1]*
AppOut-speech>*But current data indicates traffic congestion on the highway 20. Displaying an alternative itinerary, using highway 40 instead.*
AppOut-screen>[*itinerary2]*
UsrIn-speech>*I would also have to stop at location2 (gas-station, mall, friend's house, etc) on my way.*
AppOut-speech>*OK, then this would be your best itinerary:*
AppOut-screen>[*itinerary3]*
UsrOut-speech>*All right, set this itinerary.*
AppOut-speech>*Ok, take next right on…*

In order to enable these kinds of applications, several requirements need to be met by the W3C standards.:

1. **VoiceXML standard requirements for results :** EMMA is highly extensible and therefore supports rich results. However, there is no standard that mandates that a browser return these results nor is there a standard detailing how they should be rendered in the vxml environment. Moreover, we feel that certain return information should be standardized. . Some examples:

a. **Richer meaning representation for user input** : the current flat list of slot [key-value] pair (or the corresponding lattice) output isn't rich enough, we need a richer, hierarchical meaning representation that would take modifiers and semantic correlations into account. For example, the triple format of RDFS and OWL should be fully supported.
b. **Multi-source confidence** : the single confidence score doesn't convey enough information. Instead we'd need multiple confidence scores : (i) acoustic, (ii) semantic, (iii) conversational scores. In addition, it is likely that final ranking of semantic interpretations and assignment of semantic confidence is going to depend on dialog context stored in the application. Therefore, it is necessary to efficiently return multiple possible semantic interpretations along with enough information about each (e.g. acoustic and semantic) to allow the final assignment of confidence in the application.
c. **Relating semantics to the input:**
    i. Words used to determine specific semantics (when available)
    ii. Timings for words and semantics (needed to synchronize various modalities)

2. **VoiceXML requirement for vendor-specific information :** Vendor-specific parameters are critical given the current state of recognition technology and the need to experiment. However, there is no standard by which to tag such parameters as belonging to a recognition or a synthesizer resource. Nor is there a standard to tag these parameters by vendor. Finally, there is no voice-browser requirement to faithfully transmit these parameters to the speech server. Thus, vendor-specific parameters--although necessary for most real-world systems--are difficult to use in a platform-independent way.

3. **Dynamic call/interaction-flow using context**: In advanced dialogues, the static VoiceXML form-filling, FSM-based call-flow description doesn't scale. Even StateChartsXML (proposed standard for VoiceXML 3.0), which includes hierarchical state definition and parallel state composition, isn't sufficient. The fundamental limitation of state-machine based call-flow specifications is that the call-flow control is determined locally; the system is in a state *S*, an action *a* occurs, so the system transitions to state *S'*. In this type of system, all the transition logic must be coded within the current state *S* and the action *a*. Conversely, in the kind of advanced dialogues described above, semantic interpretations and call-flow decisions are contingent on a much larger context, multiple turn history, or data-driven decisions,. To model such systems as state-machines causes a state-explosion that may be unusable in even moderately-sized applications. Consequently, we believe that the decision logic must rely on the server and cannot be restricted by a standard call-flow decision model. We are entirely committed to supporting VoiceXML for the basic channeling of recognition results and prompts back and forth from the browser to the server and we'll support the state-machine based call-flow mechanisms for directed dialogues. However, we do not expect to code all the advanced dialogue call-flow logic in VoiceXML.

4. **SRGS vs. SLM :** The original srgs specification closely links ASR and semantics. But much of the past and more of the future will be based on using SLMs for recognition and separate NLU modules for semantics. We believe that W3C standards such as SRGS and SISR must adapt to these new use cases.

5. **Standard Ontologies** : Reusable domain-specific or general knowledge bases such as WordNet and its extensions, or web ontologies based on RDFS and OWL would enable unambiguous and standard coding of word senses and deeper NLU.