Conversational Architecture Requirements

Kurt Fuqua, Cambridge Mobile
kurt@cambridgemobile.com

Abstract What are the architectural requirements for a system which allows the user to interact with multiple concurrent applications using conversation? This paper examines the requirements, starting from a blank slate, for sharing the linguistic processing, linguistic data, and performing semantic routing to concurrent apps on a mobile phone.

In order to define a conversational architecture, we must start with the goals and a few definitions. A conversational system includes two key elements: a) multiple applications, b) the ability of the user to traverse the apps with ordinary conversation. This traversal should be as seamless as possible. The user should not need to know much about which app performs which function. Referenced objects made through one app should be seamlessly conveyed to another app. Most importantly, a user must *never* need to explicitly invoke a particular application. In a conversational system, the user can ask a question and the system will route the request to the appropriate application and give a natural-language answer.

Goals

- Bidirectional Natural-Language
- Cooperation
- Efficiency
- Multilingual (Reusable)
- Practical Technically & Economically

Conversations are bidirectional. The system must processes true natural-language sentences in two directions: analysis and synthesis. This processing must be done in real-time. Focus has most often been on analysis but composing sentences is also complex. The grammar must be reversible; the engines must be reversible. If it is not bidirectional, it is not a conversational system.

The second goal is that the system is cooperative. The apps must cooperatively share the processing resources of the system – the recognizer, synthesizer, lexicon, and every other level of linguistic processing and result. As we'll see the shared data must be at a high level of abstraction. Passing strings is not sufficient.

Third, the system must be extremely efficient. Multiple apps must run together on a single mobile phone with limited memory and computation. There is no room for redundant lexicons or redundant parses.

The same engines must be usable in many apps and many languages. Support for multiple languages is closely related to reusability. Ideally, a single application should be able to readily support multiple languages. The architecture must support a variety of languages, scripts, tonal languages, and unusual morphologies. It must not be hardcoded for a particular language.

Finally, the system must be practical both technically and economically. Teams that develop mobile applications are small and have limited budgets. It must be practical for a developer with no linguistic training to construct a complete system in a few days.

Examples

Here are a few examples to illustrate the challenges. "Where can I park during the show tonight?" This single sentence requires the cooperation of two applications. One app, the Scheduler, knows about the show that you have scheduled to see tonight. Another app knows about parking. The system must query the Scheduler for the location of the show, and

then pass that location and time on to the parking app. Two apps must cooperate in finding the answer.

"Hold my calls while I'm at the restaurant." Again, we need cooperation from multiple apps. One knows about the reservation, the other knows how to hold calls.

Overview

I do not assume a particular architecture in advance. I do not presume the existing VoiceXML architecture. Instead, I take a blank slate approach. What processing is needed and what data?

If the engines are to be reused for multiple languages, then we must make a clear distinction between the engine which performs the linguistic processing and the language specific data which drives that process. The language-specific information includes the lexicon and the grammar. When I use the term 'grammar', I use it in the broad sense of all facets of the language's grammar – not just the syntactic grammar. Thus the grammar would contain the information for processing morphology, phonology, prosodics, as well as syntax, and rules for translating the syntactic parses to a semantic representation. This includes everything necessary to instruct the engines about the processing for the particular loaded natural-language. By swapping this data, we change the language entirely.

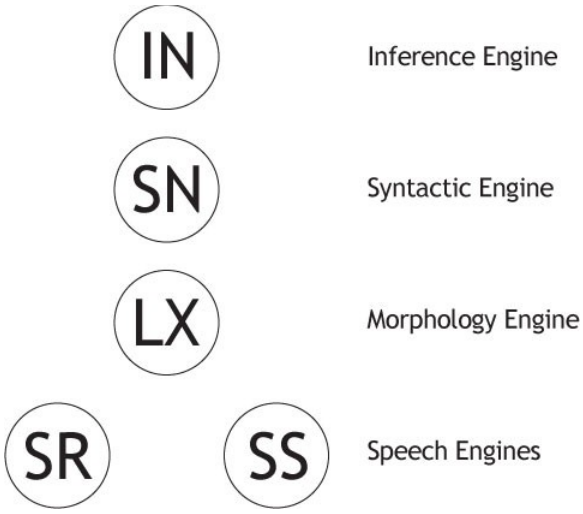
The engines carry out the processing. The engines are driven by the language grammar and lexicon. So what processing do we need for natural-language? First the process should be equally adept at processing either text or speech. When in silent mode, the user can switch to text. To support multiple languages, the system should not be limited to the Roman script or even to an alphabet. It must support abugidas and abjads – several of the 15924 scripts.

To process speech properly, we need a coherent, rigorously consistent and comprehensive system for processing speech. We need to define the phoneme set and formalize the phonology rules. This will interact with morphology. The morphology will use the lexicon. The lexical processing will also involve the orthographic processing (allographs). All of this must function in a reversible, bidirectional manner. The system will require syntactic and semantic processing.

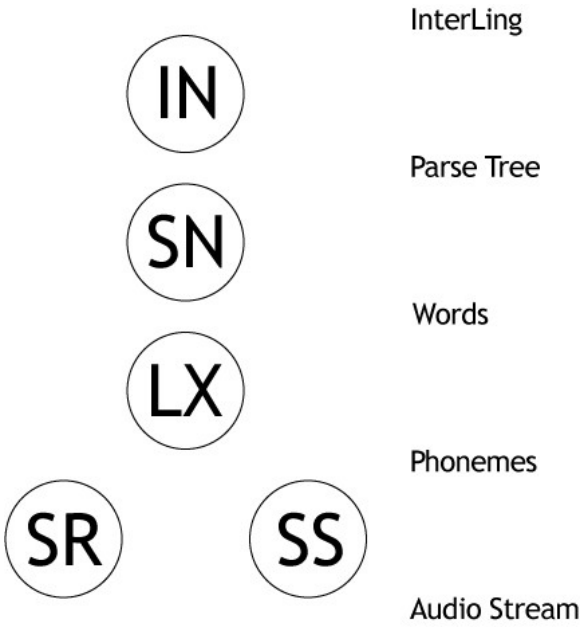
The processing should be broken into manageable modules. We should not expect a single vendor to produce each level of processing.

Where should the divisions between modules be drawn? Here is a proposal for the division. There are for levels of linguistic processing:

- speech level,
- lexical level,
- syntactic level, and
- semantic level.



This division is done such that the data passed between modules is as follows.



Bibliography

The Scalable Language IPA, Technical Reference Manual, January 2009 (http://slapi.sourceforge.net/)