VUI Visions

Continuous Automated Speech Tuning and the Return of Statistical Grammars

Roberto Pieraccini, SpeechCycle

*In this guest column, we ask designers skilled in creating Voice User Interfaces to highlight a particular aspect of VUI design inspired by actual deployments. In this issue, Roberto Pieraccini, Chief Technology Officer, **SpeechCycle** (SSN, December 2009, p. 1), Pieraccini suggests that, even in the most directed dialog interactions, a well-trained Statistical Language Model will easily outperform a rule-based grammar by allowing more varied responses, and discusses issues and solutions in using SLMs more widely. Roberto has been involved in spoken dialog technology for more than 25 years, both in research as well as in the development of commercial applications. Prior to joining SpeechCycle, Roberto was the manager of the Conversational Interaction Technology department at the **IBM** Thomas J. Watson Research Center. Before that, he led the Natural Dialog R&D group at **SpeechWorks International** (now Nuance). Earlier, he joined the Speech Research Group at **AT&T** Bell Labs and later AT&T Shannon Labs. Roberto began his career as a speech scientist with **CSELT**, the research center of the then Italian operating telephone company, after completing his doctorate in engineering from the **Universita degli Studi di Pisa**, Italy.*

Statistical grammars, commonly dubbed "SLMs" (for Statistical Language Models) by IVR practitioners, have been known to the speech research world for almost 30 years. However SLMs started to make their first steps into the IVR world only relatively recently. On the other hand, rule-based grammars, often written in an XML dialect known as SRGS (Speech Recognition Grammar Specification) have been used since the first attempts to deploy speech recognition in the early 1990s. Rule-based grammars are king for directed dialog, and only for situations where an open prompt[4] needs to be played, are SLMs brought into the equation with a lot of effort and mystique. But what the IVR world often ignores is that, even in the most directed dialog interactions, say plain YES/NO questions, a well-trained SLM will easily outperform a rule-based grammar, all other things being equal.

[4] AT&T's "How may I help you?" is the classic example of such an open prompt.

So, why are we using handcrafted, rule-based, XML grammars at all if we know that SLMs would work better? There are several reasons for that. First, building an SLM is not as easy as writing rules in XML. You need data, and it is the kind of data that you don't have when you first build a new spoken dialog system. And even if you had data—I mean recorded utterances as responses to each prompt—you would need it transcribed. On top of that you would need to provide for each transcribed training utterance—in a process called *annotation*—a semantic tag: its meaning. So while instructing transcribers to do the right thing can be straightforward, teaching annotators the correct utterance-tag mappings may be very challenging. And assuming you have your transcribed and annotated utterances, and you have cleaned all of the transcriptions and annotations to transform them into a consistent set of training data—and you need lots of them—now you have to build an SLM. How do you build an SLM?

Building an SLM requires you to have a special set of programs that do two things. The first is creating a properly called[5] *statistical language model*, in other words a way to tell the recognizer which sequences of words are legal. While rule-based grammars explicitly list all the legal sequences of words, a statistical language model does it in a ...well...statistical sense [1]. This is accomplished by computing what people of the trade call *n-grams*, which are the probabilities for any possible word—at least for all the words that appear in the training utterances—to be preceded by any possible *n-1* long sequence of words. So, if *n=3*, as it typically is in practice, the statistical language model computes, for each word, the probability of being preceded by all the possible sequences of 2 words. Thus, if you had 1,000 words in your vocabulary—and typically you need more than that for an open prompt, less than that for a directed dialog prompt—the program that computes the statistical language model has to compute 1000 (all the possible words) times 1000x1000 (all the possible pairs of words) probabilities, in other words 1 billion probabilities! Don't worry ... you may not be able to find examples of all the possible 1 billion triplets of words of your measly 1,000-word vocabulary even if you search the whole Web. But the statistical language model training program has to provide a number, the estimate of a probability, even for the most unlikely triples, like *yes maybe computer*[6]. But again, don't worry. There are programs that do that for you, and you can buy those programs; you can get them for free from some open-source packages published on the Web, or if you are versatile enough and not afraid of a little math and some algorithms, you can build them yourself. But of course, even if you had the best of programs for building the best statistical language model, you have to fiddle with a number of parameters in order to get the best out of it. And that may not be easy. But that's not all.

Building an SLM is not just about constraining the recognizer on all the possible sequences of words. In a spoken dialog system you don't need just the words that were spoken by the caller, but a semantic tag, a symbolic output from a set of *slots* that has a meaning for the call-flow at the particular prompt. For instance, if the prompt is asking a simple YES/NO question—for example "Have you paid your most recent bill?"—you want the recognizer to return either a YES or a NO. If the caller says *yes*, you want the recognizer to return YES; if the caller says "you bet" you want the recognizer to return YES; if the caller says "no way" you want the recognizer to return NO, and so on. If your training utterances are semantically annotated, on top of being transcribed, SLM builders create what is called a *semantic classifier* [2], in other words a program that takes as input the string of words recognized by the speech recognizer, and returns one out of a number of slot identifiers. Semantic classifiers are built from large samples of transcribed and annotated utterances. Again, you can buy one of these programs, or you can get it from some open-source projects, or if you feel adventurous in some non-trivial math and some non-trivial algorithms then you can build it yourself. And even if you buy it, you still have to adjust parameters and do some non-trivial tuning if you want to get the best performance.

In short, this lack of data and expertise has made using of SLMs unpopular especially for directed dialog solutions since the early days of speech IVR technology. Think about how much easier building an XML rule-based grammar is in comparison.

---

[5] Although the *statistical language model* is only a part of an SLM, the industry term SLM (Statistical Language Model) indicates the full ability to decode the meaning out of free-form utterances, or *natural language* utterance, typically the responses to an open prompt.

[6] In fact, when an uncommon triplet of words is not found in the training set, statistical language models approximate that probability using some heuristic considerations. There should not be any triplet of words with a zero probability, since that would a-priori exclude that triplet, however uncommon, for being ever recognized.

But there is another reason for SLM's lack of popularity. If you create a rule-based grammar, you can see what you did. You can immediately understand why the recognizer did not recognize the phrase "maybe yes" spoken by an undecided caller—perhaps because that phrase was not in the grammar—and you can promptly make the necessary modifications in a matter of minutes. You cannot easily do that with statistical grammars. N-grams, probabilities, and statistical classifiers are inscrutable at first sight. If something goes wrong, you need some understanding of the statistical machine learning theory behind the SLM in order to fix it. Also, the common notion—common and widely accepted in research as a logical and experimental fact—that SLMs always outperform rule-based grammars if trained on the right data, is not wholeheartedly embraced by IVR practitioners. There is confusion between the performance of an SLM in an open prompt situation, and the performance of a rule-based grammar in directed dialog. Of course the latter works better than the former, but only assuming callers always say what is in grammar, which is not always true. But this notion muddles the notion that a properly trained SLM will outperform a rule-based grammar in a directed dialog situation, especially when callers say things that are out of grammar. Yes, a well-trained SLM will have at least the same performance, and most likely outperform a corresponding rule-based grammar. So, why are we using rule-based grammars at all? As the previous paragraphs elucidate, it has nothing to do with grammar performance and everything to do with the high price of admission that SLMs entail.

And here comes the idea. What if we could provide a way to create and tune statistical grammars *automatically*, and use them for every context in a dialog, either open-prompt or directed, in place of traditionally handcrafted rule-based grammars, and do that continuously, while the application is deployed? After all, except for transcription and annotation, there is nothing that strictly requires the continuous labor of machine learning and speech scientists to build SLMs, while that is not always true for rule-based grammars. Yes, speech scientists run experiments to determine the best set of parameters, but they can create programs that run the experiments for them and decide which best selection of parameters to pick. Yes, they condition and clean the data, making sure that there are no inconsistencies in the transcriptions and annotations. But they, the machine learning and statistic speech experts, can create programs that do that for them. They can also create programs that help reduce the cost of human transcription and annotation by automating it when possible. Experienced speech scientists who are also computer scientists and machine learning experts can work on programs that tune speech grammars, rather than working on speech grammars themselves. And programs, unlike humans, can handle an abundance of data, and the speech grammars can get better and better.

What I described is the concept of automated tuning [3], or *grammar factory*, which is a service that can take a constant flow of log data from your IVR and give you, continuously, better and better grammars. How much better? Well, that depends on other factors, such as the design of the call flow, the prompts, the task, and other things that can give rise to poor grammar performance if not properly done. But while the grammar factory is looking at improving grammars, it can also flag behavior that would require the inspection of expert VUI designers and speech scientists. Will the grammars keep improving indefinitely? Certainly not. After a while, after enough data has been processed, their performance will settle on the maximum possible performance, which may not be 100% accuracy because of all of the other factors that can influence speech performance. But one thing is sure. It would be very hard, if not impossible, to reach that theoretical maximum performance using old-fashioned, handcrafted grammars tuned by hand. And if any change occurs in the application—a new prompt, a new strategy, new products, a new language—the grammar factory will, automatically and relentlessly, adjust for that and guarantee, in a short time, the attainment of the best performance for the speech recognizer. This is a new step ahead in the direction of machines that truly understand speech and continuously learn from what they hear.

References

[1] Young, S., "Talking to Machines (Statistically Speaking)," Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002), September 16-20, 2002, Denver, Colorado.

[2] Evanini, K., Suendermann, D., Pieraccini, R., "Call Classification for Automated Troubleshooting on Large Corpora," Proceedings of the 2007 IEEE ASRU Workshop, Kyoto, Japan, December 9-13, 2007

[3] Suendermann, D., Evanini, K., Liscombe, J., Hunter, P, Dayanidhi, K., Pieraccini, R., "From Rule-Based to Statistical Grammars: Continuous Improvement of Large-Scale Spoken Dialog Systems," Proceedings of the 2009 IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP 2009), Taipei, Taiwan, April 19-24, 2009