One Year of Contender: What Have We Learned about Assessing and Tuning Industrial Spoken Dialog Systems?

David Suendermann
SpeechCycle, New York, USA
david@suendermann.com

Roberto Pieraccini
ICSI, Berkeley, USA
roberto@icsi.berkeley.edu

Abstract

A lot. Since inception of Contender, a machine learning method tailored for computer-assisted decision making in industrial spoken dialog systems, it was rolled out in over 200 instances throughout our applications processing nearly 40 million calls. The net effect of this data-driven method is a significantly increased system performance gaining about 100,000 additional automated calls every month.

1 From the unwieldiness of data to the Contender process

Academic institutions involved in the research on spoken dialog systems often lack access to data for training, tuning, and testing their systems. This is simply because the majority of systems only live in laboratory environments and hardly get deployed to the live user[1]. The lack of data can result in systems not sufficiently tested, models trained on non-representative or artificial data, and systems of limited domains (usually restaurant or flight information).

On the other hand, in industrial settings, spoken dialog systems are often deployed to take over tasks of call center agents associated with potentially very large amounts of traffic. Here, we are speaking of applications which may process more than one million calls per week. Having applications log every action they take during the course of a call can provide developers with valuable data to tune and test the systems they maintain. As opposed to the academic world, often, there appears to be too much data to capture, permanently store, mine, and retrieve. Harddisks on application servers run full, log processing scripts demand too much computing capacity, database queues get stuck, queries slow down, and so on and so forth. Even if these billions and billions of log entries are eventually available for random access from a highly indexed database cluster, it is not clear what one should search for in an attempt to improve a dialog system's performance.

About a year and a half ago, we proposed a method we called Contender playing the role of a live experiment in a deployed spoken dialog system (Suendermann et al., 2010a). Conceptually, a Contender is an activity in a call flow which has an input transition and multiple output transitions (alternatives). When a call hits a Contender's input transition, a randomization is carried out to determine which alternative the call will continue with (see Figure 1). The Contender itself does not do anything else but performing the random decision during runtime. The different call flow activities and processes the individual alternatives get routed to make calls depend on the Contenders' decisions.

Say, one wants to find out which of ten possible time-out settings in an activity is optimal. This could be achieved by duplicating the activity in question ten times and setting each copy's time-out to a different value. Now, a Contender is placed whose ten alternatives get connected to the ten competing ac-

[1] One of the few exceptions to this rule is the Let's Go bus information system maintained at the Carnegie Mellon University in Pittsburgh (Raux et al., 2005).

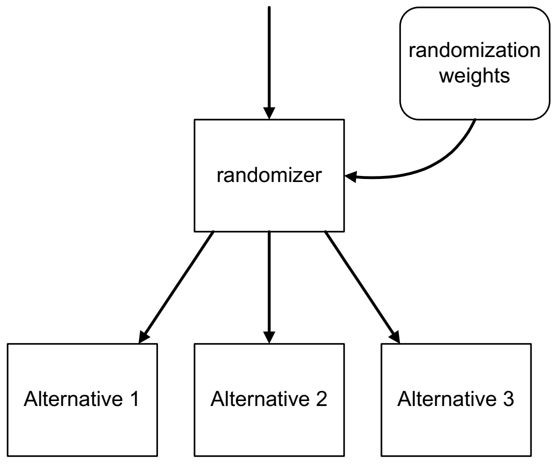

Figure 1: Contender with three alternatives.

tivities. Finally, the outbound transitions of the competing activities have to be bundled to make the rest of the application be independent of the Contender.

A Contender can be used for all sorts of experiments in dialog systems. For instance, if system designers are unsure about which of a number of prompts has more expressive power, they can implement all of them in the application and have the Contender decide at runtime which one to play. Or if it is unclear which actions to perform in which order, different strategies can be compared using a Contender. The same applies to certain parameter settings, error handling approaches, confirmation strategies, and so on. Every design aspect with one or more alternatives can be implemented by means of a Contender.

Once an application featuring Contenders starts taking live production traffic, an analysis has to be carried out, to determine which alternative results in the highest average performance. In doing so, it is crucial to implement some measure of statistical significance as, otherwise, conclusions may be misleading. If no statistical significance measure was in place, processing two calls in a two-way Contender, one routed to Alternative 1 and ending up automated and one routed to Alternative 2 ending up non-automated, could lead to the conclusion that Alternative 1's automation rate is 100% and Alternative 2's is 0. To avoid such potentially erroneous conclusions, we are using two-sample t-tests for Contenders with two alternatives and pairwise two-sample t-tests with probability normalization for more alternatives as measures of statistical significance. A more exact but computationally very expensive method was explained in (Suendermann et al., 2010a), but for the sake of performing statistical analysis with acceptable delays given the vast amount of data, we primarily use the former in production deployments.

If an alternative is found to statistically significantly outperform the other alternatives, it is deemed the winner, and it would be advisable routing most (if not all) calls to that alternative. While this hard reset maximizes performance induced by this Contender going forward, it sometimes takes quite a while before the required statistical significance is actually reached. Hence, in the time span before this hard reset, the Contender may perform suboptimally. Furthermore, even though statistical measures could indicate which alternative the likely winner is, this fact is potentially subject to change over time depending upon alterations in the caller population, the distribution of call reasons, or the application itself. For this reason, it is recommendable to keep exploring seemingly underperforming alternatives by routing a very small portion of calls to them.

The statistical model we discussed in (Suendermann et al., 2010a) presents a solution to the above listed issues. The model associates each alternative of a Contender with a weight controlling which percentage of traffic is routed down this alternative on average. As derived in (Suendermann et al., 2010a), the weight for an alternative is generated based on the probability that this alternative is the actual winner of the Contender given the available historic data. The weights are subject to regular updates computed by a statistical analysis engine that continuously analyzes the behavior of all Contenders in production deployment. In order to do so, the engine accesses the entirety of available application logs associating performance metrics, such as automation rate (the fraction of processed calls that satisfied the call reason) or average handling time (average call duration), with Contenders and their alternatives. This is relatively straightforward since the application can log call category (to tell whether a call was automated or not), call duration, the Contenders visited and the results of the randomization at each of the Contender. In Figure 2, a high-level diagram of the Contender process is shown.

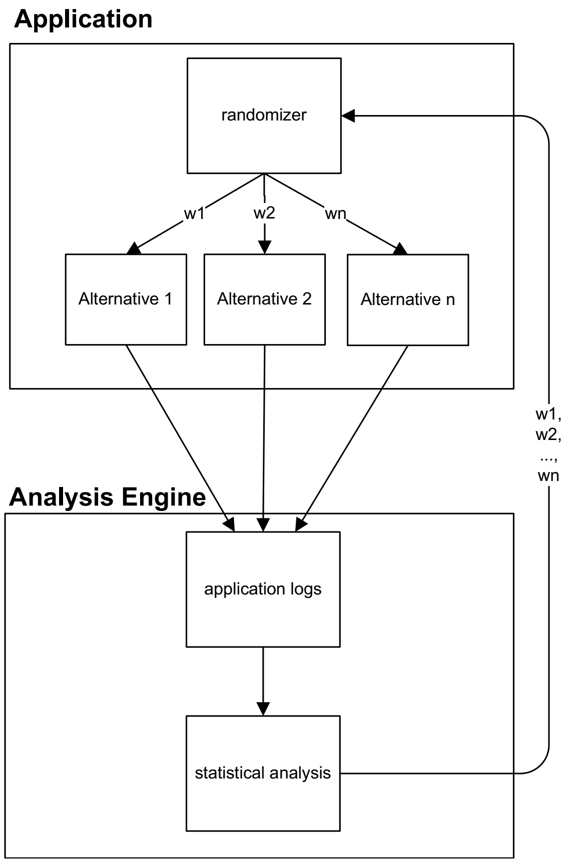

Figure 2: Contender process.

Since statistical analysis of Contenders involves data points of hundreds of thousands of calls, performance measurement needs to be based on automically derivable, i.e. objective, metrics. Popular objective metrics are automation rate, average handling time, "speech errors", retry rate, number of hang-ups or opt-outs (Suendermann et al., 2010c). There are also techniques correlating objective metrics to subjective ones in an attempt to predict user or caller experience, i.e., to evaluate interaction quality as perceived by the caller (Walker et al., 1997; Evanini et al., 2008; Möller et al., 2008). Despite the importance of making interactions as smooth and pleasant as possible, stakeholders of industrial systems often insist on using metrics directly tied to the savings generated by the deployed spoken dialog system. As we introduced in (Suendermann et al., 2010b), savings mainly depend on automation rate (A) and average handling time (T) and can be expressed by the reward

$$R = T_A A - T$$

where T_A is a trade-off factor that depends on average agent salary and hosting and telecommunication fees.

2 A snapshot of our last year's experiences

Shortly after setting the mathematical foundations of the Contender process and establishing the involved software and hardware pieces, the first Contenders were implemented in production applications. Under the close look of operations, quality assurance, engineering, speech science, as well as technical account management departments, the process underwent a number of refinement cycles. In the meantime, more and more Contenders were implemented into a variety of applications and released into production traffic. Until to date, 233 Contenders were released into production systems processing an total call volume of 39 million calls. Table 1 shows some statistics of a number of example Contenders per application. These statistics are drawn from spoken dialog systems for technical troubleshooting of cable services as discussed e.g. in (Acomb et al., 2007). Such applications assist callers fixing problems with their cable TV or Internet (such as no, slow, or intermittent connection, e-mail issues). In addition to the application and a short description of the Contender, the table shows three quantities:

- the number of calls processed by the Contender since its establishment (# calls),
- the reward difference between the highest- and lowest-performing alternative of a Contender ΔR (a high value indicates that the best-performing alternative is substantially better than the worst-performing one, that is, the Contender is very effective), and
- an estimate of the number of automated calls gained or saved per month by running the Contender ΔAt [mo^{-1}] (this value indicates the net effect of having all calls route through the best-performing alternative vs. the worst-performing one, that is, the upper bound of how many calls were gained or saved). This metric

Table 1: Statistics of example Contenders.

application	Contender	# calls	ΔAt [mo^{-1}]	ΔR
TV	problem capture	13,477,810	40,362	0.05
TV	cable box reboot order	4,322,428	28,975	0.11
TV	outage prediction	2,758,963	8,198	0.04
TV	on demand	485,300	8,123	0.17
TV	input source troubleshooting	1,162,445	3,487	0.05
TV	account lookup	9,627	3,201	0.02
Internet	troubleshooting paths I	275,248	5,568	0.02
Internet	troubleshooting paths II	1,389,489	3,530	0.01
Internet	computer monitor instruction	1,500,010	3,271	0.01
TV/Internet	opt in	6,865,929	31,764	0.05

is calculated by multiplying the observed difference in automation rate ΔA with the number of monthly calls hitting the Contender (t).

3 Conclusion

We have seen that the use of Contenders (a method to assess and tune arbitrary components of industrial spoken dialog systems) can be very beneficial in multiple respects. Applications can self-correct as soon as reliable data becomes available without additional manual analysis and intervention. Moreover, performance can increase substantially in applications implementing Contenders. Looking at only the 10 best-performing Contenders out of 233 running in our applications to-date, the number of automated calls increased by about 100,000 per month.

However, multiple Contenders that are active in the same call flow cannot always be regarded independent of each other. A routing decision made in Contender 1 earlier in the call can potentially have an impact on which decision is optimal in Contender 2 further down the call. In this respect, reward gains of Contenders installed in the same application are not necessarily additive. Not only can optimal decisions in a Contender depend on other Contenders but also on other runtime parameters such as time of the day, day of the week, geographic origin of the caller population, or the equipment used by the caller. Our current research focuses on evaluating these dependencies and accordingly optimize the way decisions are made in Contenders.

References

K. Acomb, J. Bloom, K. Dayanidhi, P. Hunter, P. Krogh, E. Levin, and R. Pieraccini. 2007. Technical Support Dialog Systems: Issues, Problems, and Solutions. In *Proc. of the HLT-NAACL*, Rochester, USA.

K. Evanini, P. Hunter, J. Liscombe, D. Suendermann, K. Dayanidhi, and R. Pieraccini:. 2008. Caller Experience: A Method for Evaluating Dialog Systems and Its Automatic Prediction. In *Proc. of the SLT*, Goa, India.

S. Möller, K. Engelbrecht, and R. Schleicher. 2008. Predicting the Quality and Usability of Spoken Dialogue Services. *Speech Communication*, 50(8-9).

A. Raux, B. Langner, D. Bohus, A. Black, and M. Eskenazi. 2005. Let's Go Public! Taking a Spoken Dialog System to the Real World. In *Proc. of the Interspeech*, Lisbon, Portugal.

D. Suendermann, J. Liscombe, and R. Pieraccini. 2010a. Contender. In *Proc. of the SLT*, Berkeley, USA.

D. Suendermann, J. Liscombe, and R. Pieraccini. 2010b. Minimally Invasive Surgery for Spoken Dialog Systems. In *Proc. of the Interspeech*, Makuhari, Japan.

D. Suendermann, J. Liscombe, R. Pieraccini, and K. Evanini. 2010c. 'How am I Doing?' A New Framework to Effectively Measure the Performance of Automated Customer Care Contact Centers. In A. Neustein, editor, *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*. Springer, New York, USA.

M. Walker, D. Litman, C. Kamm, and A. Abella. 1997. PARADISE: A Framework for Evaluating Spoken Dialogue Agents. In *Proc. of the ACL*, Madrid, Spain.