# DEPLOYING CONTENDER: EARLY LESSONS IN DATA, MEASUREMENT, AND TESTING OF MULTIPLE CALL FLOW DECISIONS

D. Suendermann, J. Liscombe, J. Bloom, G. Li, R. Pieraccini
SpeechCycle Labs
New York City, USA
{david,jackson,jonathanb,grace,roberto}@speechcycle.com

## ABSTRACT

In a recent publication [1], we laid out the mathematical foundations for an optimization technique—Contender—applicable to commercially deployed spoken dialog systems similar to what the research community would refer to as a light version of reinforcement learning. In particular, we showed how Contender respects the notion of statistical significance and outperforms the common practice in deployed systems to collect data until results appear reliable and then to draw final conclusions. While [1] was a somewhat theoretical paper focusing on the proofs of above statements and the derivation of the optimization algorithm, the present work reports on lessons we learned from live deployments in production systems. Altogether, seven Contenders were installed in five different commercial dialog systems processing 2.9 million calls over a period of three to seven months. We have found that, depending on the location of the Contender in the application's call flow and the performance difference of the competing alternatives, potentially large numbers of calls (hundreds of thousands) need to be processed to determine a winner. Furthermore, we have seen that Contender self-adapts in the event that a previously under-performing option becomes the preferred option.

## KEY WORDS

Contender, (commercial) spoken dialog systems, optimization, production deployment

## 1 Introduction

Most of today's commercial implementations of spoken dialog systems are based on the call flow paradigm, i.e., a manually designed finite state machine whose nodes can be seen as *activities* and arcs as *transitions* between these activities. Activities include:

- interactions between the system and the user to gather or provide pieces of information,
- interactions between the dialog system and other systems in local or wide area networks (backend databases, outage detection modules, line quality check, mesh-up web services, telephony network, etc.),
- any type of internal processing (running scripts, performing calculations, making random or conditional decisions, etc.).

The advantage of using a call flow over purely statistical approaches to dialog management (such as MDP [2], or POMDP [3]) is its transparency and predictability as well as its support for extremely complex tasks involving thousands of activities. Even very recent techniques such as *belief refinement* [4] or *belief recombination* [5] have not yet been shown to be applicable to real-world systems of the complexity this paper refers to (see [6] for examples of the dialog systems under consideration).

Almost every aspect and property of a call flow can be considered a variable; for example, prompt wording, dialog direction type (open, directed, or mixed), order of questions asked and/or actions taken, speech recognition thresholds (sensitivity, confirmation, rejection), etc. The question arises: What should the values of these variables be set to? The general approach—and the only feasible one for designers of complex commercial applications that require quick deployment schedules—is to set most of the variables based on the interaction designer's gut feeling, past experience, "best practices" (i.e. strategies said to work best by experts), or sometimes even random whim. Some examples of such choices are:

- The prompt is not to read "sorry, just say *yes* or *no*" because this sounds too apologetic.
- The directed dialog will not have more than five choices because cognitive science suggests that human memory cannot hold more than that many informational items.
- Ask first for the travel date and then for the departure location.
- Lower the weight of the operator grammar rule because otherwise we falsely accept noise as operator too often.
- Set the sensitivity to 0.6, the time-out to 4s, the rejection threshold to 0.2 and the confirmation threshold to 0.7.

In a recent paper [1], we presented a technique (Contender) designed to overcome the arbitrariness of such decisions.

Contender (or what the academic community would call a basic implementation of *reinforcement learning*) is a simple technique to experiment with a number of competing paths in a spoken dialog system. By randomly routing certain portions of traffic to individual paths and computing average rewards for each of the routes, the goal is to find out which one performs best according to a parameterized model of the probability density function of the reward.

From a designer's point of view, a Contender is an activity in the call flow that has one incoming transition and several outgoing transitions leading to the set of competitors for a specific Contender design. The decision of which competitor to take at runtime is made by a random generator that uses a set of weights influencing how much traffic is routed to each of the competitors (usually, the initial weights are all equal). The Contender paradigm diverges from a more traditional approach that makes a call flow change and then uses a post-mortem analysis to compare the effect on call flow performance. The drawbacks of such a traditional approach is that only one option may be considered at a time and, more importantly, the fact that the analysis is done on different user populations. If user population is not controlled then one cannot be sure that an observed increase in system performance was actually due to the change made.

Hence, under the Contender paradigm the interaction designer's decision-making process is changed from:

1. brainstorming to come up with several felicitous choices,
2. introspection about the optimal choice,
3. debating with colleagues about the optimal choice,
4. debating with the management about the optimal choice,
5. debating with the customer about the optimal choice,
6. implementing a (potentially non-optimal) choice

to:

1. brainstorming to come up with several felicitous choices,
2. implementing all choices,
3. letting the Contender paradigm decide the best choice, given a reward function.

Conservatively, one would collect as many data points as necessary to determine the statistically significant winner of a Contender instance. Depending on how much traffic the instance receives as well as how different the performance of the competitors is, this collection could take a very long time. For instance, in Section 3, we show an example where, even after collecting data for more than half a year, a final decision could not be made. As a solution to this dilemma, in [1] we showed that dynamically adjusting the weights influencing the load on every competitor according to the probability $p$ that the respective competitor is the actual winner based on the observations collected so far, is superior to waiting until statistical significance is found.

Even though in [1] we were able to demonstrate the internals of the Contender technique using paper and pencil, the publication lacked hard experimental validation on actual production data. To this end, the present paper presents results of several Contenders we have implemented into a number of commercial spoken dialog systems over the past half year.

Before going into details about the implemented Contenders, we will describe the reward function by which we evaluate Contenders in Section 2. In Section 3 we describe the Contender configurations used as well as the applications they were implemented in. Section 4 reports results and major findings.

## 2 Reward Function

The performance of commercial spoken dialog systems can be expressed in many ways, though two of the most common metrics comprise:

1. Automation rate $A$ (also referred to as task completion rate)
2. Average handling time $T$

Both metrics have a direct impact on the customer's bottom line, where a customer is a business who deploys a spoken dialog system to reduce costs incurred by employing human agents. The higher the automation rate, the higher the savings to the customer and the lower the handling time, the lower the expenses (e.g. hosting costs). These metrics are so often used because they can be calculated without human intervention by directly measuring the call outcome and call duration. It is also generally believed that these metrics correlate with user experience in that users generally prefer to complete a task and to do it in as short a time as possible. In order to evaluate the performance of our call flows, we have devised a reward function $R$ that combines both automation rate and average handling time with a trade-off rate $T_A$ [7]. The value of the trade-off rate is arrived at on a customer-by-customer basis depending on the perceived relative importance of automation rate versus handling time. Explicitly, the reward function is:

$$R = A - \frac{T}{T_A}.$$

## 3 Systems and Contenders

We will now describe the seven Contender configurations we implemented in four commercially deployed spoken dialog troubleshooting applications.

### 3.1 Cable TV troubleshooting, Provider X

Three Contenders were put into production at one of the call centers of a cable TV provider. The dialog system in question is a TV troubleshooting system able to remotely reset cable boxes, help with bad picture quality or no picture at all, poor audio quality, the channel guide, the remote control, error messages, as well as other technical problems.

C1. Call-reason disambiguation. Compare different strategies (open-ended prompt, yes/no question, directed menu) for collecting the technical problem the user is having with their cable TV.

C1a. *Please tell me the problem you are having in one short sentence.*

C1b. *Are you calling because you've lost some or all of your cable TV service?* followed by C1a) if the answer was 'no'.

C1c. C1a + *or you can say 'what are my choices'* followed by a back-up menu if the answer was 'choices'.

C1d. C1b followed by C1c if the answer was 'no'.

C2. Box unplugging instructions. Explore the impact of different troubleshooting techniques w.r.t. unplugging a cable box.

C2a. The caller is asked to unplug the cable from the back of the cable box.

C2b. The caller is asked to unplug the cable from the back of the cable box or, alternatively, from the wall.

C3. Box reboot. Explore the impact of manual or automatic cable box rebooting.

C3a. The system performs an automated cable box reboot. If it fails the caller is asked whether he or she feels comfortable doing a manual reboot. If the answer is 'no' then the caller is escalated to a human.

C3b. The system asks whether the caller feels comfortable doing a manual reboot. If he or she says 'no' the system performs an automated reboot.

### 3.2 Cable TV troubleshooting, Provider Y

The TV troubleshooting system of a second cable provider with very similar capabilities as the one described in Section 3.1.

C4. Opt-in. Explore the effect of phrasing differences in persuading the user to engage with the automated system (also referred to as *opting-in*).

C4a. *To begin troubleshooting with me, the automated agent, say 'let's start now'. [2 sec pause] Otherwise, you can say 'representative'.*

C4b. *To get started, say 'continue'. [2 sec pause] If at anytime you'd like to troubleshoot with a customer service representative, just say 'agent'.*

### 3.3 Cable TV troubleshooting, Provider Z

The TV troubleshooting system of yet another cable provider, again with very similar capabilities as the one described in Section 3.1.

C5. No-picture troubleshooting order. Explore switching the order of troubleshooting steps.

C5a. First have the caller reboot the cable box. If this does not help then ask the caller to verify that the input source is correctly set.

C5b. First verify that the input source is correctly set. If this does not help reboot the cable box.

### 3.4 Internet troubleshooting

Our Internet troubleshooting spoken dialog system can help callers resolve lost, slow, and intermittent Internet connections; fix e-mail sending and receiving problems; set up a new account, regulate parental controls, fix a missing dial tone for Voice-over-IP telephones, and the like.

C6. No-dial-tone troubleshooting. Try brute-force modem reboot versus a strategy that reboots only when necessary.

C6a. Reboot the modem.

C6b. First check the modem light pattern. If it indicates that a modem reboot would resolve the issue, reboot, otherwise, escalate the caller to a human.

### 3.5 Voice-over-IP FAQ

Our Voice-over-IP application provides answers to frequently asked questions concerning the digital phone service of a provider. Possible questions concern voicemail setup and usage, features such as call blocking, conference calls, call forwarding, or no-dial-tone troubleshooting.

C7. Call reason disambiguation. Compare different strategies (open-ended prompt versus directed menu) for collecting the calling feature of interest to the caller.

C7a. *Briefly tell me what you're calling about today.*

C7b. *There are quite a few things I can help you with. To start, just say 'voicemail' or 'calling features'. Or you can say 'help me with something else'.* Other disambiguation menus follow.

## 4 Data, Results, and Interpretation

There are a few statistics that are of interest when reporting on the results of the seven aforementioned Contender experiments. These are described below.

$T_A$ The trade-off parameter, where $T_A \to 0$ means that average handling time is considered most important and $T_A \to \infty$ means that automation rate is most important. $T_A$ is set in accordance with the customer's or account management's preference.

$time$ The number of days that a Contender has been in production.

$nTotal$ The total number of calls that a Contendered application processed in a given $time$ interval.

$nContender$ The number of calls hitting a Contender. Depending on the location of the Contender in the call flow, it will get hit more or less frequently. For example, call-reason disambiguation is usually hit by most calls, whereas no-dial-tone troubleshooting accounts for only a small percentage of Internet calls.

Table 1 shows these statistics for all Contenders introduced in Section 3. Furthermore, it shows the competitors' average rewards per Contender ($R$) as well as the winning probabilities of the respective competitors ($p$). The order of the components of $R$ and $p$ is as introduced in Section 3. E.g., for Contender C6, we read

$$(R_{\text{C6a}}, R_{\text{C6b}}) = (0.264, 0.206) \text{ and } (p_{\text{C6a}}, p_{\text{C6b}}) = (1, 0)$$

as the reward of C6a is $0.264$ and the reward of C6b is $0.206$, and the probability that C6a performs significantly better than C6b is $1.0$.

Generally, we can see that among the implemented Contenders, there are some statistically significant winners (C1, C3, C4, and C6) and some that still await a final decision (C2, C5, C7). In accordance with the usual practice in statistical hypothesis testing which is the foundation of the mathematical model underlying the estimation of $p$, we are speaking of statistical significance when the winning probability exceeds $0.95$. As discussed in our paper [1], the fact that a statistically significant reward difference is found does not solely depend on the sheer amount of data collected (C6 is found to be significant with less than $19,000$ calls, whereas C5 is still pending a decision with more than $50,000$ calls). Another important factor is the actual difference in performance of the competitors which is true for other disciplines of statistical hypothesis testing as well. When this difference is slim, many data need to be collected before a clear winner can be identified.

Let us now briefly revisit the implemented Contenders and report on the observed results in our deployed systems.

C1. The winning competitor C1b contains a yes/no question followed by an open prompt if people respond 'no' to the former. However, in case they respond 'yes', which happens about half as often, the call reason is known, and the actual troubleshooting can begin. The fact that people call about loss of service far more often than about anything else makes a yes/no question upfront more effective than the other tested competitors.

C2. The reward difference of the competitors is so marginal (a difference of just $0.003$) that a winner could not yet be determined. The alternative unplugging option (from the wall) does not seem to make much difference.

C3. Asking whether callers feel comfortable manually rebooting the cable box and acting according to their response with either manual or automatic reboot performs significantly better than a sequence of steps starting with the (often unsuccessful) automatic reboot.

C4. Even though the competitors' reward difference is small ($0.006$), we found a definite winner due to the substantial amount of collected data (almost $900,000$ calls).

C5. Apparently, whether to first reboot the cable box and then check the input source, or vice versa, does not make much of a difference. More data has to be accumulated to determine the optimum.

C6. In this case, brute force application of modem reboot outperforms an attempt at a targeted approach. The result confirms a general assumption that a flow that actively escalates callers will not perform better than a flow that does not actively escalate callers when automation rate is part of the reward function.

C7. Interestingly, in the course of the 197 days this Contender received live traffic at some point competitor C7a significantly outperformed C7b (i.e., $(p_{\text{C7a}}, p_{\text{C7b}}) = (1, 0)$). According to the paradigm explained in Section 1 we started routing $100\%$ traffic to competitor C7a. Later, however, the performance of the application started declining. At this time, though, no data were being accumulated for C7b since it was found to perform significantly worse that C7a. Due to the drop in performance of C7a, Contender's built-in re-adjustment mechanism started routing more traffic to the C7a option so that an updated comparison could be made. After collecting a sufficient amount of new C7b data, the continuous analysis will be able to conclude whether C7a is still winner or whether C7b is taking over. In this sense, we have seen that Contender acts in a self-healing mode and is able to react to dynamically changing situations.

## 5 Conclusion

We have implemented seven Contenders in five commercial spoken dialog systems that processed 2.9 million calls

| Contender | $T_A$ | $time$ | $nTotal$ | $nContender$ | $R$ | $p$ |
|---|---|---|---|---|---|---|
| C1 | 7166 | 223 | $313,210$ | $298,098$ (95.2%) | $(0.058, 0.064, 0.056, 0.060)$ | $(0, 1, 0, 0)$ |
| C2 | 7166 | 223 | $313,210$ | $43,130$ (13.8%) | $(0.222, 0.219)$ | $(0.77, 0.23)$ |
| C3 | 7166 | 223 | $313,210$ | $139,937$ (44.7%) | $(0.104, 0.116)$ | $(0, 1)$ |
| C4 | $\infty$ | 139 | $1,584,875$ | $888,240$ (56.0%) | $(0.217, 0.211)$ | $(1, 0)$ |
| C5 | 5000 | 98 | $497,923$ | $51,562$ (10.4%) | $(0.331, 0.335)$ | $(0.21, 0.79)$ |
| C6 | 5000 | 100 | $459,642$ | $18,913$ ( 4.1%) | $(0.264, 0.206)$ | $(1, 0)$ |
| C7 | $\infty$ | 197 | $49,961$ | $47,909$ (95.9%) | $(0.237, 0.233)$ | $(0.79, 0.21)$ |

Table 1. Contender statistics.

in three to seven months of deployment. The Contenders covered a variety of topics including strategies for call-reason disambiguation, troubleshooting instructions, and opt-in prompting. We have seen that, often, large amounts of data are necessary to determine the winning competitor of a Contender, and that Contender is able to dynamically react to changes of the performance distribution. We are currently working on a fully automatic analysis and update infrastructure that will allow us to incorporate substantially more Contenders into our dialog systems. Ultimately, we plan to replace human decision making by Contenders as widely as technically possible.

## References

[1] D. Suendermann and J. Liscombe and R. Pieraccini, "Contender," in *Proc. of the SLT*, Berkeley, USA, 2010.

[2] E. Levin and R. Pieraccini, "A Stochastic Model of Computer-Human Interaction for Learning Dialogue Strategies," in *Proc. of the Eurospeech*, Rhodes, Greece, 1997.

[3] S. Young, "Talking to Machines (Statistically Speaking)," in *Proc. of the ICSLP*, Denver, USA, 2002.

[4] S. Young, J. Schatzmann, K. Weilhammer, and H. Ye, "The Hidden Information State Approach to Dialog Management," in *Proc. of the ICASSP*, Hawaii, USA, 2007.

[5] J. Williams, "Incremental Partition Recombination for Efficient Tracking of Multiple Dialog States," in *Proc. of the ICASSP*, Dallas, USA, 2010.

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

[7] D. Suendermann and J. Liscombe and R. Pieraccini, "Minimally Invasive Surgery for Spoken Dialog Systems," in *Proc. of the Interspeech*, Makuhari, Japan, 2010.