Large-Scale Experiments on Data-Driven Design of Commercial Spoken Dialog Systems

D. Suendermann, J. Liscombe, J. Bloom, G. Li, R. Pieraccini

SpeechCycle Labs
New York City, USA
{david,jackson,jonathanb,grace,roberto}@speechcycle.com

Abstract

The design of commercial spoken dialog systems is most commonly based on hand-crafting call flows. Voice interaction designers write prompts, predict caller responses, set speech recognition parameters, implement interaction strategies, all based on "best design practices". Recently, we presented the mathematical framework "Contender" (similar to reinforcement learning) that allows for replacing manual decisions made during system design by data-driven soft decisions made at system run time optimizing the cumulative reward of an application. The current paper reports on the results of 26 Contenders implemented in commercial applications processing a total of about 15 million calls.

Index Terms: Contender, (commercial) spoken dialog systems, optimization, data-driven design

1. Introduction

A spoken dialog system generally consists of five main components (see Figure 1, [1]):

- speech recognition (ASR),
- spoken language understanding (SLU),
- dialog manager,
- language generation, and
- speech generation (text-to-speech synthesis, TTS).

While ASR and SLU on the one hand and language generation and TTS on the other hand serve as the interfaces between human and machine, the dialog manager can be regarded as the "brain" of the machine. It hosts system logic and knowledge, integrates with external knowledge bases and is able to perform a wide spectrum of activities, run programs, send e-mails, initiate call-backs, measure signal strength, and reboot devices to name only a few. For over two decades, dialog management has been the focus of large research endeavors such as in the ATIS [2] and Communicator [3, 4, 5] projects. Being so used to the statistical framework from the community's ground-breaking work on speech recognition [6] and understanding [7], a natural extension was to equally treat the design of the dialog as a statistical optimization problem. This was first done by [8] that modeled dialog management as a Markov decision process whose model parameters can be learned by optimizing a reward function on processed live calls. A good number of modifications to this approach were proposed in the following years, most notably the partially observable Markov decison processes [9].

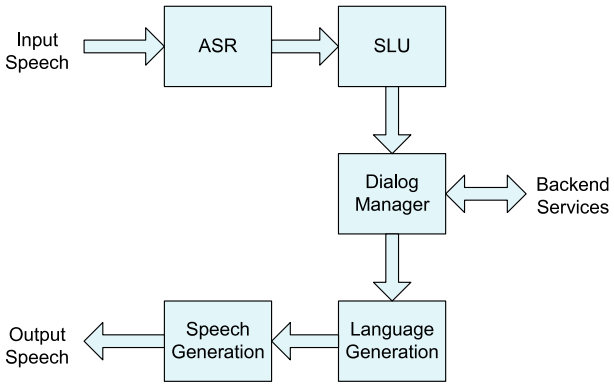

Figure 1: *General diagram of a spoken dialog system*

Admittedly, statistical approaches to dialog management are very appealing to the statisticians, mathematicians, engineers, and computer scientists in the community since they promise to learn automatically, adapt to environmental changes, and are to serve as another milestone in artifical intelligence research. However, as it stands, purely statistical dialog management has not been used in commercial applications for a number of reasons including:

- Learning directly from live calls is almost impossible since the system is bound to fail numerous times before learning a reasonable strategy by trial and error.
- The introduction of a *user simulator* [10] that navigates the dialog system instead of a human caller improves the performance of the first live call but brings up the question on how to build a user simulator. Paradoxically, a proper user simulator is either a rule-based dialog manager interacting with the statistical dialog manager or some model derived from live call statistics of a similar system. This approach requires heavy manual labor or a pre-existing dialog system, respectively.
- The beautiful mathematical framework around dialog management suffers from an immense parameter space rendering the involved computational processes intractable. Several levels of simplifying assumptions have to be made to bring the process in motion, examples are [11] or [12].
- Last but not least, spoken dialog systems based purely on statistics have not been deployed as live applications but are almost exclusively run in test environments (with the exception of some participants in the Spoken Dialogue Challenge [13]).

To build spoken dialog systems robust enough to be applicable to commercial deployments, about a decade ago, several speech technology vendors standardized a framework by which a *voice browser* is used as the interface between the standard components of a dialog system. As communication protocol between dialog manager and voice browser, the consortium specified VoiceXML [14], a language that controls which prompts to play, which language recognition and understanding models to invoke, and how to navigate between call states. VoiceXML made way to a straightforward paradigm of building spoken dialog systems in a systematic manual manner. Similar to handcrafting a decision tree, a *call flow* starts with a root activity that can, for instance, present an initial prompt to the caller. Dependening on which caller responses can be output by the SLU, a number of conditional transitions exit the root activity leading to other activities which do certain things before, again, exiting with a number of transitions. And so on and so forth. Generally, a call flow is a finite state machine whose nodes are activities and whose arcs are conditions.

The introduction of WYSIWYG tools for the generation of call flows through graphical interfaces also came with mechanisms for directly producing VoiceXML code from the graphical call flow representation. More advanced techniques produce code that is run on web servers generating VoiceXML code dynamically during application run time [15]. These tools enable quick bootstrapping of new applications. Call flows can be organized in hierarchical levels of arbitrary depth thereby allowing for highly complex applications (e.g., an Internet troubleshooting application similar to [16] built by the authors involves over 2000 activities invoking more than 10000 prerecorded system prompts and almost 1000 distinct language models and semantic classifiers).

Naturally, rapidly prototyping, building, and maintaining spoken dialog systems of this scale in a manual fashion will hardly result in optimal performance since designers mostly rely on their own experience or that of experts from the field or on heterogenious knowledge sources (analysis results from similar applications, psychology research, inference). Often, call flow portions are implemented in an ad-hoc fashion because designers know that there is no prior knowledge available for the specific topic they are working on. Answers to

- how to exactly phrase a prompt;
- whether to use an open-ended prompt, directed dialog, or a yes/no question;
- in which order questions are asked and backend queries and actions are performed;
- how to set activity parameters such as rejection and confirmation thresholds, sensitivity, time-out;
- when to escalate a caller to a human operator; etc.

are most often unknown, and, consequently, decisions are made according to the designer's gut feeling.

In a recent publication [17], we presented a technique (Contender) that is to overcome some of the guesswork that comes with designing speech applications by evaluating the performance of alternative designs on production data. Instead of settling on a single design for a given scenario, now, designers brainstorm to come up with a number of reasonable alternatives and implement all of them in the call flow[1]. At run time, a Contender activity will randomly decide which alternative to take. The random decision can be based on a set of randomization weights that make sure that, on average, a predefined amount of call traffic is routed to a given alternative. The purpose of this weighting is

- to allow for optimizing the cumulative performance of the application[2],
- to allow a certain alternative to be preferred right at the beginning of the experiment since it is expected to outperform the others, and
- to keep exploring the performance of an alternative even when it has been found to underperform (since, sometimes, the performance of alternatives can considerably change after certain events [18]).

Being a technique similar to what the academic community refers to as *reinforcement learning* (that is used by the aforementioned Markov decision processes), Contender is trying to bridge the gap between the academic and industrial approaches to building spoken dialog systems. While we laid out mathematical foundations of the Contender technique in [17] and reported on preliminary experimental results in [18], the present paper is to report on observations drawn from large-scale implementations of 26 Contenders in 10 commercial spoken dialog systems that processed about 15 million calls. Section 2 will give a brief overview of the Contenders and systems evaluated in this research. Data facts, results, and conclusions will be drawn in Section 3.

[1] This means that, after all, "best practices", design experience, and the actual act of human design of speech applications are still crucial. Human intelligence is still necessary to come up with effective design ideas. Evaluating their effectiveness is the machine's task.

[2] In [17] we showed that adjusting the alternatives' weights to reflect their respective probability of being the winner results in a higher cumulative performance than waiting until a statistically significant winner has been found.

2. Systems and Contenders

This section briefly describes dialog systems and Contenders analyzed in the present research. All the considered systems provide either technical support or FAQs for customers of cable companies. Some of the Contenders were implemented in applications of different customers. Since the behavior of a Contender can differ considerably depending on the specific customer (as examplified in Section 3), in this paper, we distinguish results by customer ID (A through E).

2.1. Cable TV troubleshooting

The cable TV troubleshooting application helps callers to resolve problems related to missing or bad picture, audio quality, remote control, channel guide, and can also refresh cable boxes.

C1. Cable box reboot order (Customers A, B)
  - a The system performs an automated reboot. If it fails, callers are asked whether they feel comfortable performing a manual reboot. If the response is 'no' they get escalated to a human agent.
  - b Callers are asked whether they feel comfortable doing a manual reboot. If they respond 'no', an automated reboot is performed.

C2. Cable box unplugging instructions (Customer A)
  - a Callers are asked to unplug their cable from the back of the box.
  - b Callers are asked to unplug their cable from the back of the box or, alternatively, from the wall.

C3. Problem capture (Customer A)
  - a *Please tell me the problem you are having in a short sentence.*
  - b *Are you calling because you lost some or all of your cable TV service?* followed by a) if the answer was 'no'.
  - c a) + *or you can say 'what are my choices'* followed by a back-up menu if the answer was 'choices'.
  - d b) followed by c) if the answer was 'no'.

C4. Input source troubleshooting (Customer B)
  - a Perform a box reboot. If it fails let the caller check whether the input source on their television is correctly set.
  - b First check for the correct input source. If it fails, perform a reboot.

C5. Outage prediction when caller has no picture (Provider B)
  - a Ask whether the problem is happening on all of the caller's multiple TVs. If the answer is 'yes' the system assumes there is an outage and escalates the call to a human agent. Otherwise, perform a box reboot.
  - b Perform a box reboot.

C6. Phrasing of opt-in prompt (Customer C)
  - a *To begin troubleshooting with me, say 'let's start now'. [2 sec pause] Otherwise, you can say 'representative'.*
  - b *To get started, say 'continue'. [2 sec pause] If at anytime you'd like to troubleshoot with a customer service representative, just say 'agent'.*

2.2. Internet troubleshooting

The Internet troubleshooting application addresses lost, slow, or intermittent Internet connections, e-mail issues, parental control settings, etc., and can also fix a missing dial tone on a Voice-over-IP phone.

C7. Order of lost Internet troubleshooting steps (Customers A, D)
- a Reboot modem, router, and computer. Then check whether the caller's Internet came back.
- b Reboot modem. Then check for success. If unsuccessful, reboot router and computer and then check again.
- c First check the modem light pattern. Only when the pattern suggests that a reboot would resolve the problem, reboot, otherwise, escalate the call to a human agent.

C8. Order of troubleshooting steps, no lights (Customers A, D, E)

The same as C7 but without Alternative c—for modems whose light patterns we are not yet researching.

C9. No-dial-tone troubleshooting of Internet modem (Provider D)
- a Reboot the modem.
- b First check the modem light pattern. Only when the pattern suggests that a reboot would resolve the problem, reboot, otherwise, escalate the call to a human agent.

C10. Account lookup (Customer D)
- a When the system knows the caller ID, it plays the message *one moment while I look up your account information* and then performs an account lookup before it goes into troubleshooting. If the caller ID is unknown, account lookup and message are skipped.
- b The system plays the message regardless of whether the caller ID is known and regardless of whether an account lookup is actually going to occur.

C11. Computer reboot instructions (Customers A, C, D, E)
- a *While your modem's getting a fresh signal, we need to shut down the* main *part of your computer; you can leave your monitor on.*
- b *While your modem's getting a fresh signal, we need to shut down your computer.*

C12. Operating system capture (Customers A, B, C, D, E)
- a *Are you running Windows or Macintosh?*
- b *Are you running Windows?*

2.3. Voice-over-IP FAQ

Our Voice-over-IP application answers questions about phone features such as voice mail, caller ID, call blocking, conference calls, and call forwarding.

C13. Call reason capture (Customer D)
- a *Briefly tell me what you're calling about today.*
- b *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'.*

C14. Caller ID disambiguation (Customer C)
- a *Do you want to block your ID or see who's calling? [pause] You can also say 'help me with something else'.*
- b *If you want to block your ID so people can't see that it's you calling, say 'block my ID'. If you want to know who's calling before you pick up the phone, say 'see who's calling'.*

3. Data, Results, and Conclusions

In order to be able to measure the performance of a spoken dialog system processing millions of calls, one has to settle on a performance metric that can be derived directly from the system logs. As motivated in [18], in this paper, we will be using a reward function that is a linar combination of automation rate (A) and handling time (T):

$$R = A - \frac{T}{T_A} \tag{1}$$

with a trade-off parameter of $T_A = 5000s$. To facilitate interpretation of the experimental results displayed in Table 1, we include the following additional statistics:

- *time*. This is the number of days the Contender was in production at the time the paper was submitted.
- *nTotal*. The number of calls processed by the application featuring the respective Contender.
- *nContender*. The number of calls processed by the respective Contender (depending on which part of the call flow the Contender is located in, this number is somewhere between 0 and *nTotal*).
- p. The probability that an Alternative is the winner based on the data collected in the course of the experiment. This probability is estimated based on the principles explained in [17]. In Table 1, R and p are displayed for the Contender Alternatives a, b, etc. in the form

$$(R_{\mathrm{a}}, R_{\mathrm{b}}, \ldots) \quad \text{and} \quad (p_{\mathrm{a}}, p_{\mathrm{b}}, \ldots). \tag{2}$$

In the table, performance values of alternatives that were found to be statistically significantly different from competing alternatives are identified by printing their winning probabilities p in **bold**. We consider a result *statistically significant* when p approaches either 0 or 1 with a significance level of 0.05. This means that if a Contender has two alternatives, both, p_a and p_b will be found either significant or not. When there are more than two alternatives, however, it is possible that the probabilities of some alternatives are found to be significant while others are not. E.g., Alternative c of C7A was found to significantly underperform while the contention between Alternatives a and b is not decided yet.

Whether alternatives are found to be statistically significant winners or losers depends on the observed performance differences and on the amount of data collected. E.g., C6 features a clear winner (Alternative b) even though its performance is only slightly higher than Alternative a (0.170 vs. 0.174), however, the sheer amount of data analyzed (almost 1.7 millon calls) showed the result to be significant nonetheless. Also C12D has a clear winner (Alternative b) although only 1343 calls hit the Contender. Here, the performance difference was found to be substantial (0.150 vs. 0.214).

In contrast to our very first step towards implementing Contenders in commercial spoken dialog systems [18], the present work attempted to evaluate the effect of population-specific Contenders by rolling out the same Contender to different customers and treating them as separate experiments. For example, C1 has shown significant results for Customer A while there is only a marginal performance difference for Customer B. In the case of C12, almost all customers (marginally) tend to Alternative a (A, B, C, and E), however, the only significant result is that of Customer D which clearly finds Alternative b to be the winner. Even more interesting, for C11, all participating customers (A, C, D, E) (marginally) tend towards Alternative a. When we now take the data collected for Customer A and limit the analysis to only one of the 16 call centers the customer operates, we find that, for this specific caller population, Alternative b significantly outperforms a (C11A').

Table 1: Contender statistics.

Contender	Customer	*time* $[d]$	*nTotal*	*nContender*	R	p
C1	A	376	644,819	281,897 (44%)	(0.117, 0.110)	(**1**, **0**)
	B	153	613,007	240,380 (39%)	(0.274, 0.273)	(0.74, 0.26)
C2	A	376	644,819	89,451 (14%)	(0.229, 0.230)	(0.40, 0.60)
C3	A	376	644,819	615,162 (95%)	(0.057, 0.062, 0.054, 0.057)	(**0**, **1**, **0**, **0**)
C4	B	134	657,535	71,015 (11%)	(0.331, 0.332)	(0.48, 0.52)
C5	B	153	613,007	76,634 (13%)	(0.159, 0.309)	(**0**, **1**)
C6	C	255	2,927,289	1,684,134 (58%)	(0.170, 0.174)	(**0**, **1**)
C7	A	287	2,302,878	193,589 (8%)	(0.275, 0.279, 0.250)	(0.37, 0.63, **0**)
	D	267	1,176,073	126,714 (11%)	(0.234, 0.238, 0.194)	(0.33, 0.67, **0**)
C8	A	30	229,421	12,828 (6%)	(0.251, 0.261)	(0.11, 0.89)
	D	126	1,053,522	45,873 (5%)	(0.310, 0.313)	(0.30, 0.70)
	E	224	81,784	42,606 (52%)	(0.352, 0.389)	(0.08, 0.92)
C9	D	255	1,176,073	26,316 (2%)	(0.267, 0.204)	(**1**, **0**)
C10	D	126	526,761	88,650 (17%)	(0.158, 0.164)	(**0.01**, **0.99**)
C11	A	30	229,421	24,654 (11%)	(0.415, 0.406)	(0.86, 0.14)
	A'	30	23,807	2,306 (10%)	(0.407, 0.452)	(**0.05**, **0.95**)
	C	15	193,037	47,967 (25%)	(0.212, 0.207)	(0.92, 0.08)
	D	23	69,058	13,416 (19%)	(0.406, 0.395)	(0.90, 0.10)
	E	20	4,407	1,302 (30%)	(0.389, 0.374)	(0.71, 0.29)
C12	A	30	229,421	6,384 (3%)	(0.136, 0.130)	(0.69, 0.31)
	B	90	90,070	3,920 (4%)	(0.035, 0.028)	(0.69, 0.31)
	C	15	193,037	5,781 (3%)	(0.042, 0.025)	(0.94, 0.06)
	D	23	69,058	1,343 (2%)	(0.150, 0.214)	(**0.01**, **0.99**)
	E	20	4,407	283 (6%)	(0.075, 0.070)	(0.54, 0.46)
C13	D	365	100,815	83,711 (83%)	(0.193, 0.237)	(**0.02**, **0.98**)
C14	C	15	4,958	210 (4%)	(0.037, 0.024)	(0.67, 0.33)

The observation that results of Contender experimentation seem to (sometimes) depend on certain external parameters (in our case we showed that one such parameter is the caller population) motivates us to perform deeper analysis into which parameters (time of the day, day of the week, call reason, caller expertise, etc.) influence optimal decisions of Contenders. When decisions will be made at run time depending on the specific parameter situation of the call, the notions of "Contender" and "data-driven design" become fuzzy. The original intention of introducing Contenders was to help designers make optimal decisions rather than to rely on best practices or gut feeling. Going forward, designers will have to accept that, often, there will not be clear answers of the type "b is the winner" anymore, but unsatisfying statements such as "b is the winner unless it is for Customer A or D or for Customer B's Call Center 13, however, only when it is a weekend and the callers are from Area Code 212 and have not called already during the last 48 hours ...".

4. References

[1] W. Minker and S. Bennacef, *Speech and Human-Machine Dialog*. New York, USA: Springer, 2004.

[2] C. Hemphill, J. Godfrey, and G. Doddington, "The ATIS Spoken Language Systems Pilot Corpus," in *Proc. of the Workshop on Speech and Natural Language*, Hidden Valley, USA, 1990.

[3] A. Rudnicky, E. Thayer, F. Constantinides, C. Tchou, R. Shern, K. Lenzo, W. Xu, and A. Oh, "Creating Natural Dialogs in the Carnegie Mellon Communicator System," in *Proc. of the Eurospeech*, Budapest, Hungary, 1999.

[4] A. Potamianos, E. Ammicht, and J. Kuo, "Dialogue Management in the Bell Labs Communicator System," in *Proc. of the ICSLP*, Beijing, China, 2000.

[5] M. Walker, J. Aberdeen, and G. Sanders, *2001 Communicator Evaluation*. Philadelphia, USA: Linguistic Data Consortium, 2003.

[6] L. Rabiner and R. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, USA: Prentice Hall, 1978.

[7] D. Klatt, "Review of the DARPA Speech Understanding Project," *Journal of the Acoustical Society of America*, vol. 62, no. 4, 1977.

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

[9] S. Young, "Using POMDPs for Dialog Management," in *Proc. of the SLT*, Palm Beach, Aruba, 2006.

[10] K. Scheffler and S. Young, "Automatic Learning of Dialogue Strategy Using Dialogue Simulation and Reinforcement Learning," in *Proc. of the HLT*, San Diego, USA, 2002.

[11] 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.

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

[13] A. Black and M. Eskenazi, "The Spoken Dialogue Challenge," in *Proc. of the SIGdial Workshop on Discourse and Dialogue*, London, UK, 2009.

[14] S. McGlashan, D. Burnett, J. Carter, P. Danielsen, J. Ferrans, A. Hunt, B. Lucas, B. Porter, K. Rehor, and S. Tryphonas, "VoiceXML 2.0. W3C Recommendation," `http://www.w3.org/TR/2004/REC-voicexml20-20040316`, 2004.

[15] R. Pieraccini and D. Suendermann, "Experiments in automatic grammar localization of commercial spoken dialog systems," in *Multilingual Natural Language Applications: From Theory to Practice*, D. Bikel and I. Zitouni, Eds. Upper Saddle River, USA: Prentice Hall, 2011.

[16] 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.

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

[18] D. Suendermann and J. Liscombe and J. Bloom and G. Li and R. Pieraccini, "Deploying Contender: Early Lessons in Data, Measurement, and Testing of Multiple Call Flow Decisions," in *Proc. of the HCI*, Washington, USA, 2011.