Dialog Trajectory Analysis

J.H. Wright, A. Abella and A.L. Gorin

AT&T Labs – Research
{jwright, abella, algor}@research.att.com

Abstract

Spoken dialog systems are becoming increasingly common in deployed services. These systems are not perfect, and are often deployed "open-loop" — lacking in systematic procedures for diagnosing problems and for making improvements. In order to target improvements where they will have the biggest impact two things are needed: first, methods and tools for detailed analysis of a data feed of call logs and customer audio; second, an interactive tool for presenting an intuitive view of the results to those responsible for the application. In this paper we discuss the paradigm and an implementation through which we are able to close the loop between system execution and system evolution by providing an empirical dialog trajectory analysis represented via a stochastic finite state machine. Novel graph analysis algorithms are introduced for change detection, compression and pruning for display, based on user-interface objectives.

1. Introduction

Dialog systems exist in a variety of instantiations, each allowing the user a different interaction medium. There exist the touch-tone dialog systems that accept only keypad input, requiring a user to select from a predefined set of options that may or may not reflect the user's problem. There also exist directed dialog systems that allow speech input but greatly constrain what the user can say. Quite often these systems do not differ greatly from traditional touch-tone systems. The most flexible of dialog systems enable user initiative, allowing the user to describe their problem in unconstrained fluent speech, shifting the burden from the user to the system [1,2]. A method for monitoring the user-system interaction is required regardless of the type of dialog system.

There are two traditional ways to monitor deployed spoken dialog systems. First, call monitoring enables operations personnel to dial in and listen to a series of calls made to the system. This has the advantage that the listeners can hear the customers' speech and assess the experience from the customers' viewpoint. However, the sample of calls is bound to be very small, it may be unrepresentative because of the timing of dial-in sessions, and the listeners' assessments will be subjective. Second, summary reports are normally available on a daily or weekly basis. Typically these give a breakdown of the overall outcomes of all the calls into the system, including the numbers of service completions, transfers to agents, and hang-ups. But they are usually too coarse-grained to be of diagnostic value when some parts of the system are not performing well. The same is true of the usability measures widely used for spoken dialog systems [3,4].

In this paper we describe new technology for automatically "listening" to all calls, providing fine-grained analysis and diagnosis, real-time system evaluation and business intelligence. At the core of this technology is an algorithm for creating an empirical call- flow that enables the analysis and evaluation of the system with respect to the call-flow specification.

The call-flow specifies the actions the application should perform based on the user input and possibly pertinent external information about the user that is retrieved from a database. The call flow is organized into sub-dialogs, each of which may involve a series of turns with the user.

The empirical call-flow lays out the actual user's path through the call flow. It is generated for many users over many interactions thus creating a representation of the system behavior.

2. Visualization of dialogs

2.1 Data feed

In order to close the loop between a deployed system and its analysis, a regular data feed is required, with provision for the operations personnel (who are responsible for the system) to view the results. This is illustrated in Figure 1. Dialog systems are equipped with the ability to record key pieces of information to a call-log. What information is recorded varies greatly from application to application. Section 2.3 describes the challenges associated with key pieces of information missing from the call-log. Currently the call-logs generated by the production system are sent via ftp to a database that organizes the call-logs and the customer audio. The analysis and visualization tools mine the database and generate results that are viewable on a web server through an interactive web tool that the

operations personnel use to determine recommendations for changes to the system.

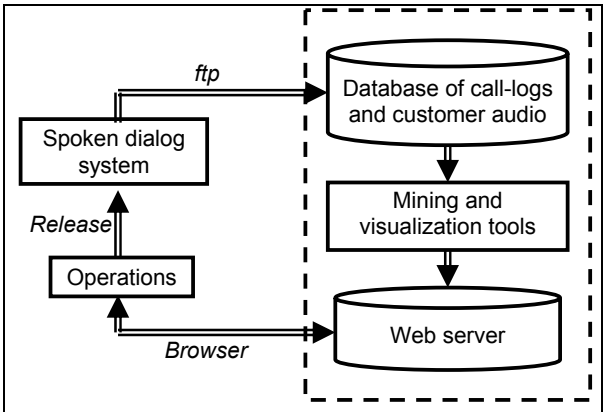

Figure 1: Closing the loop between system and analysis.

2.2 Hot-Spot Detection

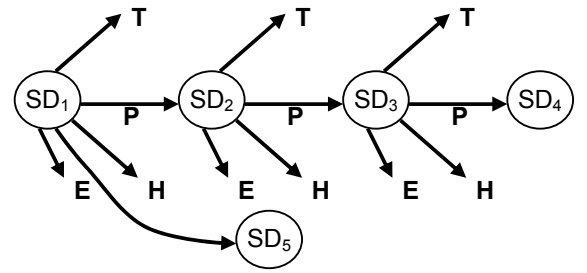

Figure 2: Finding sub-dialog hot-spots.

Some systems require long dialogs with customers, extending over many turns, especially where the dialog is tightly directed. These are best partitioned into sub-dialogs using the call-flow as a guide. Each sub-dialog must end in exactly one of four possible ways: proceed to next (P), transfer to agent or to another system (T), end-call by system (E), hang-up by caller (H), see Figure 2. Not all of these may be implemented for all systems or sub-dialogs. The first three are determined by the call-flow, the last by the caller.

We are especially concerned with sub-dialogs that have a relatively low rate of proceeding to the next sub-dialog. If there is a large number of transfers from a particular sub-dialog we want to be able to zoom in and discover the precise reasons for this. Such *hot-spots* are good candidates for system improvements via changes to call-flow and models, implemented through system releases.

2.3 Hidden variables

Consider the question of how to represent distributions of dialogs. The structure of a dialog (in general) is a sequence of sub-dialogs, each a sequence of turns, each a set of attributes with values. By projecting a sub-dialog onto a sequence of one of these attributes to form a **dialog trajectory** we are then able to sort, count, and view as a stochastic finite state machine (FSM). If we project onto two attributes we obtain two interleaved sequences, which can be represented on the nodes and arcs of the FSM.

Ideally a node of the FSM would correspond with a unique state within the call-flow. If the logging is sufficiently comprehensive then this can be achieved with little or no manual intervention. In practice, call-logs are often incomplete, correspondence between call-flow and call-log is often *implicit*, and the outcome of the call (transfer to agent or another IVR, system end-call, user hang-up without completion, service completion) is not explicitly logged. The call-flow state and call-outcome must then be treated as hidden variables, and algorithms developed to infer these from observations.

3. Methods and tools

3.1 From call-logs to FSM

The four essential steps in visualizing the dialogs are as follows:

1. Either extract directly from each call-log (if possible) or infer (if needed) interleaved sequences of two chosen attributes that characterize the system state and user response.
2. Append the call outcome (transfer to agent or another IVR, system end-call, user hang-up without completion, service completion). This completes the dialog trajectories.
3. Sort, count, and convert the dialog trajectories into a minimized stochastic FSM [5], with the system state represented on the nodes and the user response on the arcs. The call outcomes are encoded in the terminal nodes.
4. Plot the FSM using a visualization package. For this we used the Graphviz package [6].

This can be done daily, weekly, monthly, or depending on need. Examples are shown in section 4.

3.2 Change detection

The behavior of a system is likely to change over time, as a result of system releases, user behavior and traffic routing into the system. Detecting and highlighting these changes requires two refinements. The first step is to detect the changes, but we want to focus on the independent sources of any change — consequent changes are distracting. Here we consider only changes relative to a chosen reference period, rather than trends or cycles.

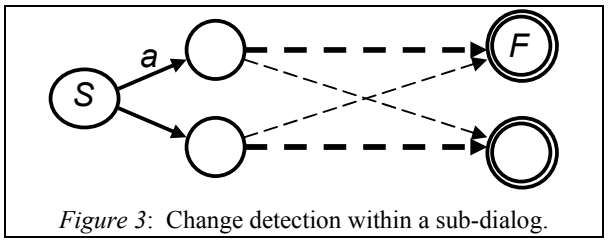

Figure 3: Change detection within a sub-dialog.

Because we are most interested in call outcomes, we first test each sub-dialog terminal node F (see Figure 3) to find out whether P(F) has significantly increased or decreased. If so, we search among the arcs for contributory influences. Arc a (from state S) is significant for terminal node F if

- P($a|S$) has significantly changed in the same direction as F (increased or decreased), and
- P($F|S,a$) is significantly greater than P($F|S,\bar{a}$)

All of these tests are done using 2x2 contingency tables, using exact methods for small counts and the standard chi-square approximation for large counts.

These conditions ensure that an arc is significant for a terminal node because of a local change at state S, and not just because more trajectories are entering state S as a result of other changes upstream. For display on the Web, significant changes (arcs and nodes) are represented in color.

3.3 Sub-graph compression

The sub-dialog trajectory FSMs can be quite large and it is useful to be able to compress them to make the significant changes more prominent. This is not the same as pruning, where we delete arcs and nodes according to some criterion. Instead, sub-graphs containing no significant changes are compressed into a single arc represented by a dashed line.

Figure 4(a) shows part of a sub-dialog FSM, containing two significant arcs (bold arrows). The first step is to label certain nodes as visible as follows:

- Start and all terminal nodes
- Nodes connected by significant arcs

All significant arcs are retained, and dashed arcs are created wherever there are two or more arcs between visible nodes. The result is shown in Figure 4(b). The trajectory count associated with a dashed arc is accumulated over the trajectories that pass through its source and destination, excluding trajectories also passing through source nodes of other arcs (including dashed arcs) that terminate at the same destination.

3.4 Web Interface

An interactive web tool was created to enable the operations personnel to view the results of the analysis.

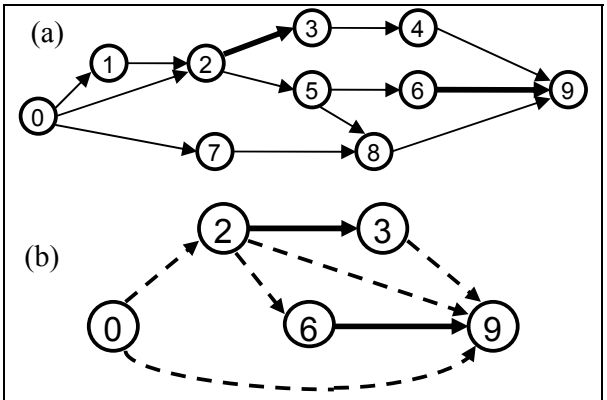

Figure 4: Part of FSM, (a) uncompressed, (b) compressed.

The web tool allows a user to view weekly dialog trajectories. There are multiple views of the trajectories. When a user selects a week they see an overview of the empirical call-flow for the entire application, as shown in Figure 5. Each node on this graph represents a sub-dialog. The web tool enables the user to click on the node and see the graph representing the sub-dialogs as shown in Figure 6. The nodes on each sub-dialog graph represent the grammars that were active. Clicking on the grammar nodes reveals statistics about the grammar, including the numbers of rejections, recognitions, and silences. It also displays the top 6 recognition results.

Each sub-dialog page reveals two options to the user. If there has been a significant change from the reference to the current period then a button appears that enables the user to view only the portion of the graph that has changed significantly from the reference period. Also associated with each page is an option for an expanded view of the graph that displays the user utterances on the arcs.

4. Trouble ticket application

4.1 Description of the application

This is a dialog system for creating a trouble ticket that details a problem with a telephone or data circuit. Coarse-grained statistics are available that describe overall numbers of transfers and hang-ups, but with no indication of where or why. A daily data feed is received that contains the recognizer call-logs. These call-logs contain turn information including a time stamp, grammar name, recognizer status and result. Missing from the call-logs are the prompts that were played, the results of any touch-tone input, and the call outcome. The state within the call-flow at each turn, and the call-outcome, are hidden variables that have to be inferred from the sequence of grammars and user responses.

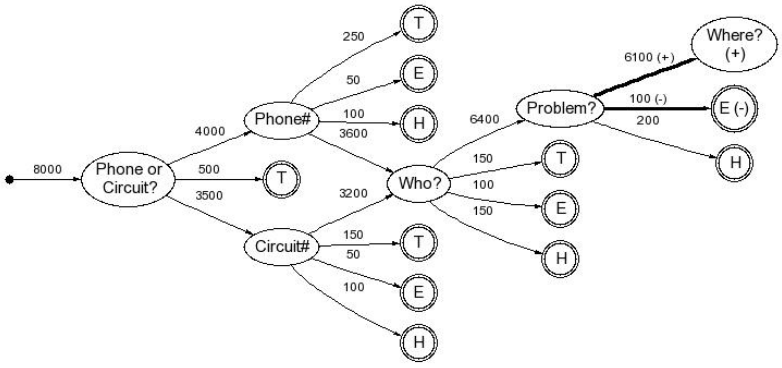

Figure 5: Sub-dialogs overview.

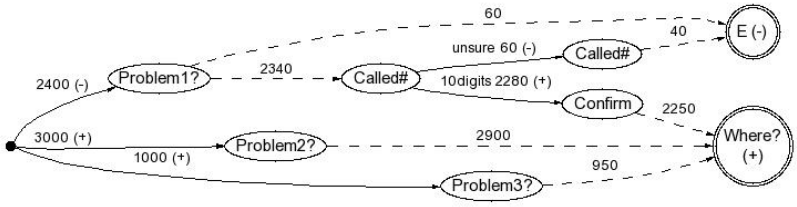

Figure 6: Problem? sub-dialog, compressed.

4.2 Results

Figure 5 shows a sub-dialog overview plot, similar to Figure 2. This and detailed plots for all the sub-dialogs are generated automatically from the call-logs received each day. The six sub-dialogs are as follows:

- Phone or Circuit?: Determine whether problem is with a phone or a data circuit.
- Phone#: Get and confirm the trouble phone number.
- Circuit#: Get and confirm the trouble circuit number.
- Who?: Get contact information.
- Problem?: Get details of the problem.
- Where?: Get location information.

The numbers on the arcs are ficticious, but they illustrate the approach. The proportion of calls reaching the Where? sub-dialog has significantly increased compared with a reference period (indicated by the + on the arc and in the node), and the proportion of end-calls for this sub-dialog has decreased (indicated by the –).

To understand this in greater detail we turn to the trajectory plot for this sub-dialog, Figure 6. The Where? sub-dialog is treated as a terminal node for this sub-dialog. The full plot actually contains more than 17,000 arcs, so we use the compression procedure (section 3.3) to highlight the arcs that significantly influence the terminal nodes (section 3.2). Hang-ups are not displayed because they have not significantly changed. We see two changes that explain the observed shift. First at the start node, fewer calls are routed through the Problem1? question, with a greater number being routed through two alternatives. This reflects a call-flow change. Second, in eliciting a Called# a greater proportion of users are providing a valid 10-digit number, with a smaller proportion unsure. Together these explain the significant changes in the end-call and Where? terminal nodes.

5. Conclusions

We have described a new procedure for visualization of dialog trajectories, with the ability to zoom in to pinpoint hot spots. Machine and customer channels are represented on the nodes and arcs of an FSM. Novel graph algorithms are introduced for inferring hidden variables, change detection, and subgraph compression.

6. Acknowledgements

The authors would like to thank David Lerner, Tina Grobe and Raymond Murry for their support and enthusiasm. Also David Kapilow for help with data extraction, and John Ellson for help with the Graphviz package.

7. References

[1] A.L.Gorin, G.Riccardi and J.H.Wright, "How May I Help You?", Speech Communication, 23: 113-127, 1997.

[2] A.L.Gorin, A.Abella, T.Alonso, G.Riccardi and J.H.Wright, "Automated natural spoken dialog", IEEE Computer Magazine, 35(4):51-56, 2002.

[3] L.B.Larsen, "Assessment of spoken dialogue system usability – what are we really measuring?", Proc. Eurospeech-03, Geneva, pp. 1945-1948, 2003.

[4] M.Walker, D.Litman, C.Kamm and A.Abella, "PARADISE: A framework for evaluating spoken dialog systems", Proc. 35th Annual Meeting of ACL, pp.271-280, 1997.

[5] M.Mohri, F.C.Pereira and M.Riley, "Weighted finite-state transducers in speech recognition", Computer Speech and Language, 16(1):69-88, 2002.

[6] E.R.Gansner and S.C.North, "An open graph visualization system and its applications to software engineering", Software – Practice and Experience, 30(1):1203-1233, 2000.