# Interactive Visualization of Human-Machine Dialogs

*Jeremy Wright, David Kapilow and Alicia Abella*

AT&T Labs – Research, Florham Park, New Jersey, U.S.A.
{jwright,dak,abella}@research.att.com

10.21437/Interspeech.2005-784

## Abstract

Automated spoken dialog systems require systematic procedures for evaluating performance and diagnosing problems. We present an interactive tool that provides graphical views of how callers navigate through such systems, enabling fine-grained analysis for system evaluation and business intelligence. The input is a feed of call-logs. The output is an empirical dialog trajectory analysis represented as stochastic finite state machines, accessible via the web. Complexity is managed by an automatic tokenization procedure that hides fine details until needed. Users can generate selective views of parts of the dialog at high resolution (with access to call data), or zoom out to a summary. The tool provides dialog system developers with all the information they need from a single source, and is in use with directed-dialog and natural-language applications.

## 1. Introduction

Dialog systems support different user response models. Touch-tone systems accept only keypad input, requiring a caller to select from a predefined set of options that may or may not reflect their problem. Directed-dialog systems allow speech input but greatly constrain what the caller can say. Natural language systems enable user initiative, allowing the caller to describe their problem in unconstrained fluent speech, shifting the burden of understanding over to the system [1,2].

Some dialog systems handle tens of millions of calls per annum. Questions that naturally arise are:

- Where are things going well?
- Where are things going poorly, and why?
- How does performance vary over time or dialog strategy?

All types of dialog systems require a method for monitoring the caller-system interaction. In this paper we view a dialog as a series of prompts and responses, culminating in a call *disposition*. Possible dispositions are transfer to agent, redirect to another system, end-call by system, and hang-up by caller. Each of these may occur either before or after completion of a service. We describe an interactive tool for analyzing all calls, providing fine-grained analysis and diagnosis, for 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 specification call-flow. If the call-flow is complex, it may be organized into sub-dialogs, each of which may involve a series of turns with the caller. The empirical call-flow lays out the callers' actual paths through the dialog system. It is generated for many calls over many interactions thus creating a representation of the system behavior.

The empirical call-flow can be viewed at different resolutions. A coarse-grained view enables the user to see the overall flow, a fine-grained view reveals full detail and provides access to individual calls. A statistical comparison between two sets of dialogs for different times using the empirical call-flow can reveal significant changes that may be attributable to either system changes, network routing of calls into the system, or caller behavior patterns.

## 2. System overview

### 2.1. Dialog trajectories

Consider the question of how to visualize large numbers of dialogs. The structure of a dialog is a sequence of turns, each of which is a set of attributes with values, usually including the prompt name, the ASR grammar, status and result, and perhaps caller-intent classification and dialog state. This complex structure is hard to visualize as a whole, so we simplify it by projecting a dialog onto sequences of values of selected attributes. Because a dialog is an interaction we project the dialog onto two interleaved sequences to form a *dialog trajectory***.** One sequence represents the machine actions, normally using the prompt name. The other represents the caller responses, which may be silence, ASR rejection or result, or caller-intent. Named-entity values, such as names and phone numbers, and caller-intents (semantic classes) are automatically tokenized. The dialog trajectory is terminated by the call *disposition.*

We then cluster these interleaved sequences over many dialogs and represent them as a set of minimized finite-state machines (FSMs [3]), with the machine actions on the nodes and user responses on the arcs, together with arc occurrence counts for a particular set of call-logs. These FSMs are then rendered visually using a graph-plotting package [4,5].

### 2.2. Data feed

The process starts with a data feed from the dialog system. For each call the system creates a log containing key pieces of data about the call. The logs contain a wealth of information: the prompts heard by the caller, the caller's response at each turn — ASR or DTMF result, and possibly a caller-intent classification — along with ASR and barge-in status, the call disposition, timing information, the audio, etc. Once the call-log data has arrived and been archived from the production system, an abstraction layer parses the logs for the information needed to create the dialog trajectories.

### 2.3. Sub-dialogs

For complex dialog systems the full dialog trajectory FSM may be too large to be viewed conveniently. If the call-flow specification is partitioned into sub-dialogs then the empirical call-flow may be similarly partitioned. Each sub-dialog has a functional description such as "get and confirm a telephone number". Ideally the name of each sub-dialog entered would be logged. If not, then simple rules for defining sub-dialogs can be created that automatically label the data, typically based

on prompt names. For each sub-dialog we generate a trajectory FSM as described above. In addition we generate an *overview* FSM, which shows how the sub-dialogs are interconnected and how they terminate.

### 2.4. Web tool

An interactive web tool enables the user to explore the data for a chosen date-range, which could be a single day or week, or all the days for a particular release of the system. All the FSMs and tabular information are generated on demand. Client-side image maps enable the user to explore and obtain information using the mouse, for example:

- Responses to selected prompts
- All dialogs that pass through selected nodes
- Detailed views of parts of the dialog, selected by clicking on boundary nodes
- Summaries of individual calls that have particular properties.

The user may also listen to customer audio and view manual transcriptions of caller utterances where these exist.

The tool is used by operations managers and user experience personnel responsible for a range of deployed human-machine dialog systems. At the simplest level, the user can obtain a summary of the numbers of calls terminating in the various dispositions, and the number of service completions, for a chosen period of time. Comparisons over time, and alerts generated in response to significant changes are also possible. At a deeper level, the user can obtain information about what the callers are doing at each point in the call flow, and how these actions lead to either a successful or unsuccessful termination of the call. This information is vital for planning and monitoring improvements to the systems, and the evolution of the tool has been guided by the needs of its users.

### 2.5. Application portability

The tool can be ported to different applications with a minimal amount of effort. At the lowest level, a data abstraction layer extracts the required prompts and responses from the logs created by the operations platform. These logs tend to differ significantly with the platform, but are similar for different applications running on the same platform.

For directed dialog applications that are partitioned into sub-dialogs, a table is created that has lists of prompt numbers from sub-dialog entry and exit. With the aid of the application's call flow documentation, this table can be created in about an hour for most applications.

If the system logs do not generate named-entity values, the system provides an application dependent routine to convert the user responses into appropriate named-entities based on the response and the prompts both before and after the response. This tuning to the application will often take a few hours.

## 3. Interactive views of complex dialogs

### 3.1. The dialog entropy problem

Even with tokenization of named entities and semantic intents, the longer the dialog the more likely it is to be unique. For example, for a sample of 26k calls for the trouble ticket application described in section 4.1 there are 14k distinct dialog trajectories of which 13k occur exactly once. The corresponding minimized FSM consists of 64k nodes and 75k arcs. One potential way to alleviate this problem is to take just the most frequent dialog trajectories, base the FSM on these, and ignore the rest. It turns out, however, that not only is the coverage then very partial, it is also unrepresentative. The most frequent dialog trajectories are not necessarily the most typical, they may just be the shortest ones. The expressed preference of the users is also to start from a summary view of all the dialogs, and then to recover fine details for a subset as required.

A better solution is to extend the tokenization to fragments of dialog, allowing more trajectories to merge, and hence reducing the size of the FSM. The tokenization needs to be entirely automatic, carried out to the point where the resulting full-coverage FSM will be sufficiently small to be easily viewable, and all suppressed detail retained for subsequent recovery on demand via the web tool.

### 3.2. Trajectory clustering and tokenization

A dialog trajectory is essentially a string of symbols, so a natural way to proceed is to define a distortion measure and implement a clustering strategy. Because all the web pages are generated on demand, the clustering needs to be fast. For this reason we use a sparse dynamic-programming algorithm to find the maximum-length common subsequence [6] for two trajectories. An edit distance is inferred by comparing each trajectory with this subsequence. This procedure is typically from one to two orders of magnitude faster than the usual Levenshtein edit distance method for the dialogs we are dealing with. Clustering is agglomerative and proceeds until it is predicted that the minimized FSM will be sufficiently small.

Each cluster is represented by a "reduced" trajectory from which all actual elements can be obtained by insertions of turns and/or substitutions of caller responses. For example, the following three trajectories (interleaved prompts $p$ and responses $r$)

$$\begin{aligned} &p_1\ r_{11}\ p_2\ r_{12}\ p_3\ r_{13}\ p_4 \\ &p_1\ r_{21}\ p_2\ r_{22}\ p_x\ r_{2x}\ p_3\ r_{23}\ p_4 \\ &p_1\ r_{31}\ p_2\ r_{32}\ p_y\ r_{3y}\ p_z\ r_{3z}\ p_3\ r_{33}\ p_4 \end{aligned} \tag{1}$$

can all be obtained from the reduced trajectory

$$p_1\ r_{11}\ p_2\ \text{(more)}\ p_3\ r_{13}\ p_4 \tag{2}$$

where the special symbol "(more)" indicates that there are turns to be inserted for some trajectories. When clusters are merged, the reduced trajectory of the new cluster is updated so as to preserve this property. The choice of clusters to merge is governed by a cost function which depends on the trajectory frequencies and the extent of suppressed detail.

After clustering, each actual trajectory is compared with the reduced trajectory for the respective cluster, and any turns covered by the "(more)" symbol are extracted and appended for possible subsequent recovery. The three trajectories in equation (1) then become

$$\begin{aligned} &p_1\ r_{11}\ p_2\ r_{12}\ p_3\ r_{13}\ p_4\ \ \# \\ &p_1\ r_{21}\ p_2\ \text{(more)}\ p_3\ r_{23}\ p_4\ \ \#\ r_{22}\ p_x\ p_{2x} \\ &p_1\ r_{31}\ p_2\ \text{(more)}\ p_3\ r_{33}\ p_4\ \ \#\ r_{32}\ p_y\ r_{3y}\ p_z\ r_{3z} \end{aligned} \tag{3}$$

The minimized FSM is then formed from the prompt sequences of the reduced trajectories, ensuring compactness, and the user responses are added to the arcs. Where multiple

user responses connect the same pair of nodes (e.g. $r_{11}, r_{21}, r_{31}$ in (3)), these become multiple labels on a single arc. With tokenization of named entities and semantic intents, the number of these is typically small. After this automatic tokenization of dialog fragments, the FSM referred to in section 3.1 which has 64k nodes and 75k arcs is reduced to 124 nodes and 210 arcs, and represents visually the high-level structure of the dialogs. Even without any sub-dialog partitioning, this full-dialog FSM is now viewable and navigable, and provides a summary view of all the calls.

The hidden substrings (after the '#' symbol in equation (3)) are carried forward until needed. The user may choose to expand that part of the graph in order to see them, or may request the response distribution for prompt $p_2$ which would include $r_{12}, r_{22}$ and $r_{32}$.

### 3.3. Generating selective views

Typically the user interacts with the tool to generate a series of selective views of the dialogs, in order to answer questions of the following form:

1. What do callers do *here*?
2. How do callers get from *here* to *there*?
3. What are *typical* and *atypical* dialogs?
4. What has changed between *then* and *now*?

All of these require a calendar for data selection, and questions of type 4 require a second calendar. Questions of types 1 and 2 generally require the FSM to be augmented in order to display the hidden substrings, and compressed to hide the unwanted details elsewhere. The compression procedure is described in [7]. FSM augmentation makes use of existing nodes and arcs as far as possible, but with constraints to preserve acyclicity.

Questions of type 3 require a probability model of the dialogs (see section 3.4) and the ability to select a range. For the latter we use a dual-slider, so that the user can select (for example) the top 20%, the bottom half, or the middle third. Questions of type 4 require a change detection procedure. This is focused on the terminal nodes of each sub-dialog and the causal changes within the sub-dialog that help to explain the changes at the terminals, and is described in [7].

In general, a selective view involves augmenting, pruning and compressing the FSM, and there may be several iterations of this. It is essential to minimize the resulting FSM again, so that a tidy view is presented to the user.

### 3.4. Trajectory probability estimator

The simplest answer to the question "What is the probability of a dialog?" is the maximum likelihood estimate: the frequency of this trajectory relative to all dialogs. This suffers from the shortcoming described in section 3.1, where most of the dialog trajectories are singletons and therefore receive the same probability. The representation of a set of dialogs as an FSM enables a first-order Markov model to be used in a natural way. Thus the estimated probability of a prompt-response sequence becomes

$$\tilde{P}(p_1\ r_1 \cdots p_{L-1}\ r_{L-1}\ p_L) = P(p_1)\prod_{i=1}^{L-1} P(r_i, p_{i+1}|p_i)$$

Dialogs that traverse frequently-used parts of the FSM then receive a higher probability than ones that don't, even when they have the same raw frequency. These probability estimates are ranked and used for the range selection for questions of type 3, described in section 3.3.

## 4. Applications

### 4.1. Trouble ticket application

The visualization tool is now deployed for several applications. One of these is a directed-dialog system through which business customers having problems with telephone or data circuits can create trouble tickets. A daily data feed of call-logs is received, containing turn information including a time stamp, prompt name, ASR grammar name, ASR status and result, as well as call-level information such as the calling number, date/time, and call disposition. The sub-dialogs for this application are described in [7].

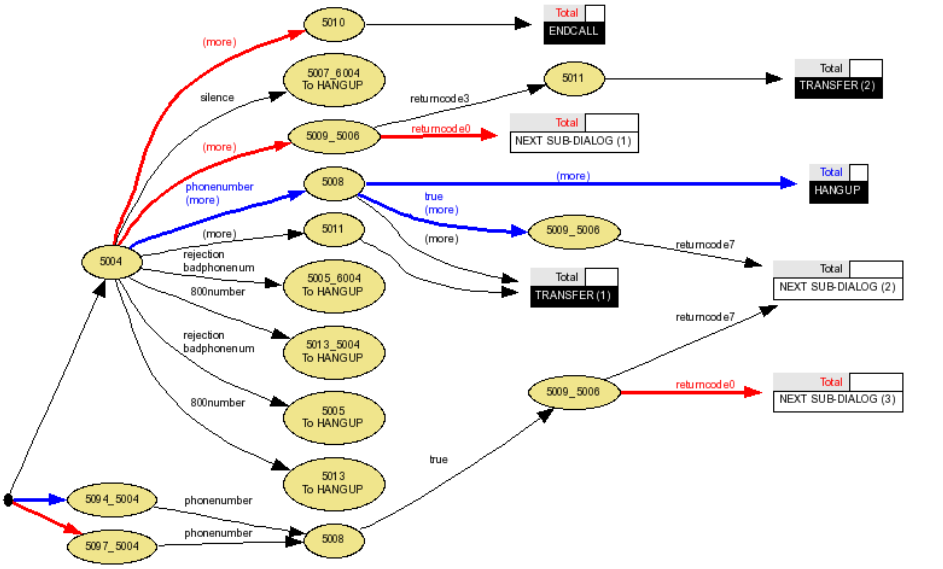


Figure 1: "Get telephone number" sub-dialog for trouble-ticket application.

The sub-dialog shown in Figure 1 acquires and confirms a telephone number that the customer is having trouble with. All counts on arcs and nodes have been omitted. Where prompt numbers are concatenated within a node, the first part is a message. Prompt 5004 requests the number, prompt 5008 confirms it. The caller responses (and ASR status) may include silence, rejection, "phonenumber" or "800number" tokens, "badphonenum" meaning a number rejected by the system as invalid, and "true" meaning an affirmative confirmation ("yes" or equivalent). Prompt 5006 tests the number against a database, the response being a return-code, so this constitutes a turn with a database rather than with the caller. There are terminal nodes for hang-up, end-call, and

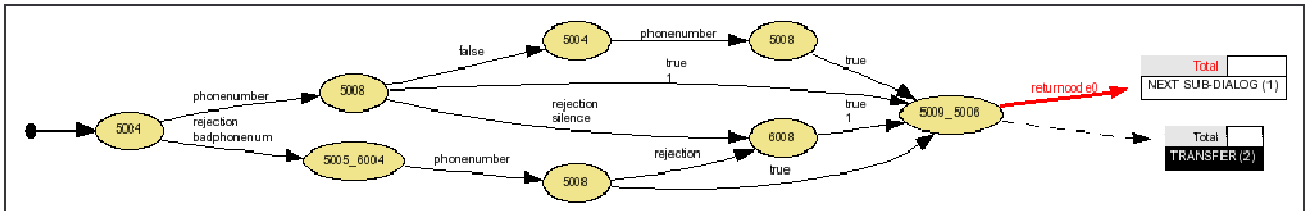


Figure 2: Expanded arc within "Get telephone number" sub-dialog for trouble-ticket application.

two transfer queues for human agents, and for three possible next sub-dialogs. Colors indicate significant changes relative to a comparison period, red for an increase, blue for a decrease.

The arcs labeled "(more)" contain hidden substrings as described in section 3.2. The arc from node 5004 to 5009_5006 in Figure 1 is expanded in Figure 2. The top 95% (ranked as described in section 3.3) of these trajectories are shown. Giving the user control in this way helps to manage complexity — the remaining 5% are mainly one-off trajectories that more than double the size of the FSM, and add more confusion than useful information. We are especially interested in trajectories near the top of the ranking that contain some persistent problems such as ASR rejection, misrecognition, or silence from the caller. These are good places to look for opportunities to significantly improve system performance.

### 4.2. Customer care application

This is a natural-language call-routing application using a "How May I Help You?" open-ended initial prompt [1], running on the AT&T VoiceTone® spoken dialog platform. A classifier generates a semantic intent which is used to label the arcs. There is typically a much higher initial fan-out for such applications than for those involving directed dialog. A natural way to deal with this is to partition the trajectory plot into pages, sorted by disposition node in decreasing order of frequency. Figure 3 shows the first of three pages for the initial greeting part of this application. Access to the trajectory plots is integrated into the "Daily News" system [8], and the user can interact with these in the manner described above.

## 5. Conclusions

We have described a method and tool for visualization of human-machine dialogs, using FSMs to represent machine actions on the nodes and user responses on the arcs. Call dispositions (transfer, redirect, end-call or hang-up) are represented as final nodes of the FSM. The FSMs are rendered graphically on an interactive web server, which allows the user to see both a high-level view of the system and to zoom in to see fine details. Complexity is managed by an automatic tokenization procedure that hides fine details until needed. We have demonstrated the utility of these approaches for both directed-dialog and natural-language applications.

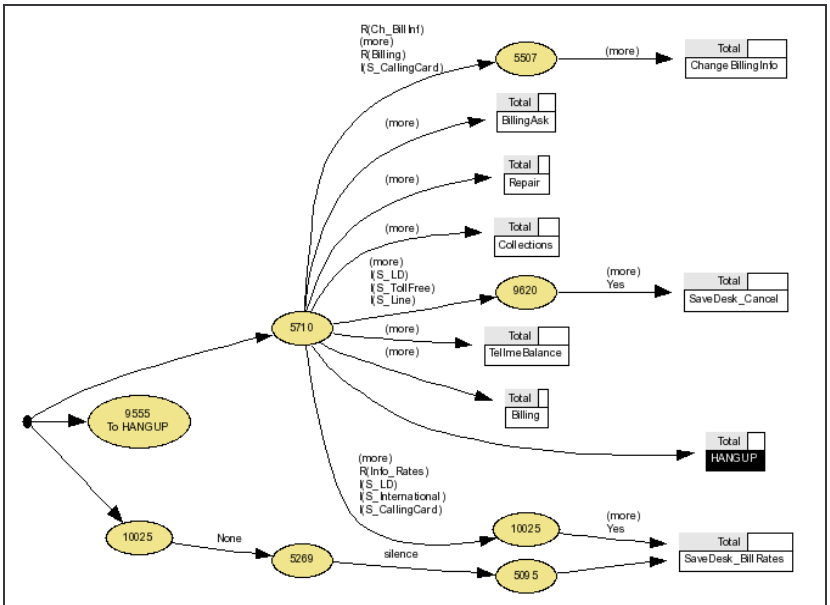


Figure 3: First page of greeting sub-dialog for customer care application.

## 6. Acknowledgements

The authors would like to thank David Lerner, Maria Alvarez-Ryan, Bob Costa and Tina Grobe for their support and enthusiasm. Also John Ellson and Emden Gansner 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*, vol. 23, 1997, pp. 113-127.

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

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

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

[5] AT&T Labs – Research, Graphviz, [Online]. Available: http://www.research.att.com/sw/tools/graphviz

[6] D.Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, 1997.

[7] A.Abella, J.H.Wright and A.L.Gorin, "Dialog Trajectory Analysis", *Proc. ICASSP-04*, Montreal, 2004.

[8] S.Douglas, D.Agarwal et al, "Mining customer care dialogs for 'Daily News'", *Proc. ICSLP*, Korea, 2004.