

Mining Customer Care Dialogs for “Daily News”

Shona Douglas, Deepak Agarwal, Tirso Alonso, Robert Bell, Mazin Rahim, Deborah F. Swayne, Chris Volinsky

Abstract—As large-scale deployments of spoken dialog systems in call centers become more common, a wealth of information is gathered about the call center business as well as the operation of these systems from their daily logs. This paper describes the “VoiceTone Daily News” data mining tool for analyzing this information and presenting it in a readily comprehensible and customizable form suitable for use by anyone from system designers to call center businesses. Relevant business and dialog features are extracted from the speech logs of caller-system interactions and tracked by a trend analysis algorithm. We describe novel techniques for generating alerts on multiple data streams while avoiding redundant “knock-on” alerts. Some initial experiments with automated measures of dialog success are described as possible additional features to track. Features that move outside their expected bounds on a given day generate headlines as part of a web site generated completely automatically from each day’s logs. A “drill-down” facility allows headlines to be investigated all the way to viewing logs of individual interactions behind the headline and listening to the audio for individual turns.

Index Terms—speech mining, data mining, spoken dialog systems, dialog success, business intelligence, trend analysis, alerts, speech recognition.

I. INTRODUCTION

Over the past decade, there has been a significant rise in the number of speech services used across a wide variety of market segments to provide electronic contact mechanisms between people and organizations. Companies such as AT&T, Nuance and Scansoft have deployed large-scale spoken dialog systems that receive several million calls per year and produce terabytes of data in the form of operation logs and speech audio [1], [2], [3]. Traditionally, this data is used for standard reporting, such as providing statistics on the number of calls per day, percentage of completed calls, hang-up rate, etc. A small sample of this data is typically transcribed and annotated by human labelers in order to compute system accuracy and improve the underlying speech recognition and understanding models.

As the volume of data grows and the number of deployments rises, we face the major technical challenge of mining these interactive spoken dialogs to quickly extract critical application information and business intelligence. The basic question is whether one can explain not only what is happening but why. For example, when a spoken dialog system is first deployed, it is sometimes the case that a large percentage of callers hang up or insist on speaking to a live agent. It is not only important to discover how many people are declining to communicate with the system, but to learn why. Can caller behavior be explained by geographical location, age, gender, or previous interactions? Such discoveries can help to improve both user experience and system performance. Even better, can these

discoveries be generated automatically by algorithms which automatically examine callers’ profiles and behavior?

Mining spoken dialog data for the purpose of extracting business intelligence is a new research area that has not been well explored in the past. It poses several interesting and new theoretical challenges including (a) identifying relevant features of the dialogs, (b) creating robust statistical and learning methods for capturing trends, discovering patterns automatically, and identifying critical service and business information, and (c) applying language generation and visualization methods for presentation of key information.

In this paper, we describe the *VoiceTone Daily News* [4], a data mining tool that supplements AT&T’s VoiceTone[®] system by *automatically* extracting critical service information and business intelligence from customer care dialogs. VoiceTone is a call center service that creates, deploys and hosts natural-language spoken dialog applications for large business customers [5]. The *Daily News* uses the spoken dialog interaction logs along with meta information including customer profiles to automatically detect interesting and unexpected patterns and to present them in a daily web-based newsletter intended to resemble on-line news sites such as CNN.com or BBC.co.uk.

This paper describes the statistical and linguistic methods used for generating the *Daily News*. Section II describes the AT&T VoiceTone system. Section III describes the architecture of the “Daily News” system. In Section IV we briefly describe feature extraction and storage. The statistical methods by which we track and identify trends are described in section V. Section VI describes how the web site presents the analyses to the user. In Section VII, we provide a summary of this paper.

II. AT&T VOICETONE

VoiceTone, announced by AT&T in 2003, is a hosted solution that helps contact centers to reduce their operation cost, increase their service revenue and, most of all, improve on the user experience of traditional IVR (Interactive Voice Response) systems. The AT&T VoiceTone spoken dialog applications support mixed initiative dialog providing users the flexibility to speak naturally. They represent a new generation of automated contact center services that greet the customer with the open-ended prompt “*How May I Help You?*”.

VoiceTone is powered by advanced speech technologies including Automatic Speech Recognition (ASR), Spoken Language Understanding (SLU), Dialog Management (DM), Spoken Language Generation (SLG), and Text-To-Speech Synthesis (TTS). Speech is recognized by AT&T’s Watson ASR engine which uses continuous-density hidden Markov models and statistical language models. The Watson engine supports any-vocabulary recognition and is based on advanced acoustic

System: Hello, this is Barney health care, how may I help you?
User: My medicine ran out last week → *Request(Refill)*
System: Would you like to refill your prescription?
User: yes uh my prescription number is PB14720. → *Yes, pres_num=PB14720*
System: Okay, I have ordered your new refill. Is there anything else I can help you with today?
User: No thanks → *No*
System: Thank you for calling Barney Health Care. Goodbye.

Fig. 1. An example of a VoiceTone dialog from a healthcare application

and network optimization algorithms for robust and efficient speech recognition [6]. For unrestricted input speech, the Watson engine performs at about 78% word accuracy. The recognized speech is processed by the SLU and is tagged for the presence of domain-relevant “named entities” using finite state automata and mapped to one of a set of **call types** using BoosTexter, a member of the AdaBoost family of large-margin classifiers [7]. The DM processes the output of the NLU along with the dialog history and determines an action. This is converted into words and then played out to the user by the TTS Natural Voices engine [8]. An example of a VoiceTone dialog is shown in Fig. 1, illustrating the system’s classifications (*Request(Refill)*, *Yes*, *No*) and named-entity extraction (*pres_num=PB14720*).

The system logs each dialog interaction to a database as a set of time-stamped events generated by the various system components. Among the recorded events are prompts played, recognition output, classifier output, and call routing behavior. An XML document is generated daily, consisting of a sequence of dialog elements, one for each call, each consisting of a time-ordered sequence of the logged events. All applications built on the VoiceTone platform share the same log file format, but differ in their particular ASR grammars, SLU models, dialog design, and final routing destinations.

III. VOICETONE DAILY NEWS SYSTEM

The overall system architecture for the Daily News is shown in Fig. 2. Each night, the previous day’s dialog logs are processed to extract dialog and turn level features, which are loaded into a database. (A turn is a segment of a dialog, defined in the next section.) A set of statistical analyses is run on these features, to update cumulative statistics and to track and identify trends.

The numeric results of the statistical analyses are generated in a general format which allows them to be flexibly structured into a layered web site which presents high-level natural language descriptions in addition to detailed tabular and graphical reports. The web site integrates this information with dynamic query access to the feature database and to the SeeCalls visualization tool.

IV. FEATURE EXTRACTION

The first part of the feature extraction process is the application of XSLT templates (XSLT [9] is a tree-oriented transformation language for transmuting instances of XML) to

divide the stream of events in each dialog into a sequence of turns. A turn consists of a system prompt, the user response as recognized by the system, and any records associated with the system’s processing of that response. Many kinds of analysis and evaluation relevant to spoken language systems naturally operate at the turn level.

From this representation, a set of features is extracted for each dialog and for each turn within a dialog. The extraction specification for each feature is represented declaratively as a fragment of XSLT.

Currently extracted dialog features include the originating telephone number for the call, the number of turns, the length of the call, any final routing destination, and what is termed the **final actionable call type**. This is the last call type the classifier obtained before routing or hangup which is also “actionable” in the sense of not being vague or uninformative (if none are actionable, the last is used). Turn features include the prompt and speech recognition output, the classifier call type and the turn duration.

Feature values are deposited in the database for run-time use, and also sent to the statistical analysis module. We use AT&T’s Daytona™ database technology [10], which combines huge capacity and fast operation with simplicity of administration and the flexibility of using either its own 4GL Cymbal, SQL, or normal UNIX text processing tools on the database tables, which are simply flat files.

V. ANALYSIS AND ALERTING

A. Simple Feature Alerts

Alerts are designed to identify abnormal occurrences in data streams. All features can be tracked by looking at the prevalence of the values of a given feature over a specified time period. We use standard statistical process control techniques [11] that generate an alert if the most recent prevalence of a value/feature pair is outside the expected interval based on data from an appropriate historic period.

More specifically, at time t , let y_{it} represent the percentage of calls that have value i at time period t . For a given window of n time periods, calculate the mean percentage of calls with value i over that time period, $m = \sum_{j=(t-n)}^{t-1} y_{ij}/n$. Similarly calculate the variance over that time period, s_i^2 . Then the expected interval for y_t , the percentage at time t is:

$$y_t \in (m_i - 3s_i, m_i + 3s_i). \quad (1)$$

Anything outside of this range is considered out-of-process, and generates an alert. Features with percentages outside a one standard deviation range are considered minor alerts.

Features of many types can be tracked via this mechanism. The main ones we look at are the values for the **final actionable call type**, since this alerts us to changes in customer behavior (e.g., a sharp increase in cancelled orders or customer complaints). Integer and real-valued features such as number of turns and call durations are binned so that we may alert them in a similar fashion.

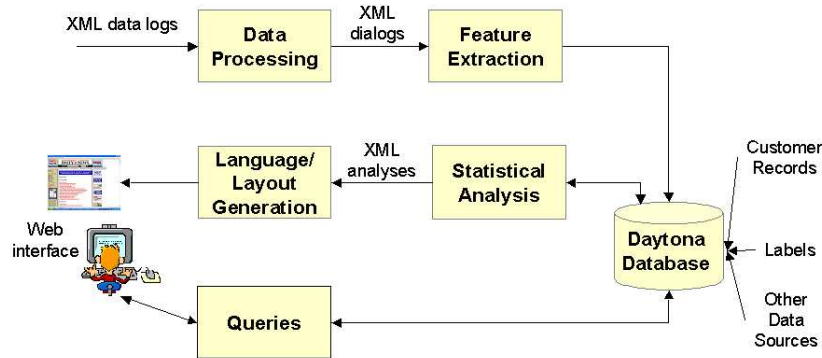


Fig. 2. VoiceTone Daily News system architecture.

B. Alerts over multiple data streams

For this application, we need to process **multiple data streams**. For instance, each of the roughly 100 final actionable class values or the roughly 50 routing destinations in one application can be considered a single data stream. The performance of simple threshold-based alerting methods deteriorate when monitoring multiple streams due to the well known multiple testing problem. Performing a statistical test a large number of times will lead to false positives even if nothing is amiss. Methods to ameliorate this situation for static data have a vast literature in statistics both from a classical and a Bayesian perspective. However, none of these techniques have been explored to detect changes in multiple data streams. The method we use here (see [12] and references therein for more details) addresses this issue.

Another problem with alerting procedures designed for single streams can be seen in the following situation: assume a systems problem results in a 50% increase in calls which go to an operator. That feature value will alert correctly, but the increase in that feature value necessarily causes decreases in all of the other values for the same feature, resulting in multiple alerts, when there was only one main cause.

In the next three subsections, we will describe the multiple testing technique for the static case, then the algorithm, called **hbmix**, which we use for data streams, and finally experimental evidence for the superiority of **hbmix**.

C. Multiple test procedure for the static case

Let δ_{it} denote the deviation from the past of the i^{th} stream at time point t . Possible ways to compute δ_{it} are discussed in Section V-D. Let s_{it}^2 denote a variance measure associated with δ_{it} computed using observations prior to t . Assume $\delta_{it} \sim N(\Delta_{it}, \sigma_i^2 s_{it}^2)$. It will often be necessary to transform the original observations for this normality assumption to be more reasonable. In the VoiceTone Daily News project, the original proportions were transformed using $\text{Sin}^{-1}\sqrt{y}$ and re-normalized to sum to unity. In general, a power transformation provides a rich parametric class to choose from. At time t we want to test the multiple hypotheses $\Delta_{it} = 0; i = 1, \dots, K$. Assume with probability p_t that Δ_{it} is exactly 0 (i.e., the null hypothesis is true) and with probability $1 - p_t$ it follows $N(\mu_t, \tau_t^2)$ distribution. In our

application, we always choose $\mu_t = 0$ for each t . We note that the transformation applied to the original proportions along with “shrinkage” toward 0 has worked well in reducing redundant alerts discussed earlier. To complete the Bayesian specification, we need appropriate priors on the parameters $(p_t, \tau_t^2, \sigma_t^2)$. The prior for p_t is $\pi(p_t) \propto p_t^\alpha$ ($\alpha \geq 0$). The joint prior for (σ_t^2, τ_t^2) is $\pi((\sigma_t^2, \tau_t^2)) = \pi(\sigma_t^2)\pi(\tau_t^2|\sigma_t^2) = (1/\sigma_t^2)s_{0t}^2/(s_{0t}^2 + \tau_t^2/\sigma_t^2)(1/\sigma_t^2)$ where $s_{0t}^2 = K/\sum_i(1/s_{it}^2)$ (harmonic mean).

Posterior of hyperparameters: The marginal distribution of δ_{it} is explicitly obtained by marginalizing over Δ_{it} and is $p_t N(0, \sigma_t^2 s_{it}^2) + (1 - p_t) N(0, \sigma_t^2 s_{it}^2 + \tau_t^2)$. Hence, by Bayes theorem, the posterior of $\Theta_t = (p_t, \tau_t^2, \sigma_t^2)$ is proportional to the product of the marginal likelihood times the prior and denoted by $\pi(\Theta_t|\delta_t)$. This is not available in closed form since the constant of proportionality involves computing an intractable 3 dimensional integral.

Posterior of Δ_{it} 's: The posterior of Δ_{it} conditional on Θ_t is a mixture with point mass Q_{it} at zero and with probability $1 - Q_{it} \sim N(m_{it}^p, v_{it}^p)$ where

$$Q_{it} = \frac{p_t \phi(\delta_{it}/\sigma_t s_{it}) / [p_t \phi(\delta_{it}/\sigma_t s_{it}) + (1 - p_t) \phi(\delta_{it}/\sqrt{\sigma_t^2 s_{it}^2 + \tau_t^2})]}{1}$$

where $\phi(z) = \exp(-z^2/2)/\sqrt{2\pi}$

$$m_{it}^p = B_{it} \delta_{it} \text{ where } B_{it} = \tau_t^2 / (\tau_t^2 + \sigma_t^2 s_{it}^2) \text{ and } v_{it}^p = \sigma_t^2 s_{it}^2 B_{it}$$

The unconditional posterior is obtained by integrating this conditional density with respect to $\pi(\Theta_t|\delta_t)$. We accomplish this by using Gauss Hermite quadrature. (See [12] for more details).

Declaring change using a decision theoretic approach:

Deciding on a change involves making a compromise between false positive and false negative rates. Dropping suffixes i and t , let $L(a, \Delta)$ denote the loss incurred for action a (which can take values C indicative of change and N indicative of no change) when the truth is Δ . The Bayes rule is the action which minimizes posterior expected loss. For a 0–1 loss function, this corresponds to the rule which declares a change if $Q \leq .5$. However, this might not always be sensible. For instance, if Q is as high as .8 and with probability .2 the posterior shows extreme change we are better off taking action C . Similarly, a low value of Q if complemented with mild

change should encourage action N . Motivated by this, we use the loss function

$$\begin{aligned} L(C, \Delta) &= I_{\Delta=0} \\ L(N, \Delta) &= c_1 I_{\Delta>0} + c_2 I_{\Delta<0} \end{aligned} \quad (2)$$

In (2), if $c_1 = c_2 = c$, increases and decreases are penalized in a symmetric fashion. The average of $L(C, \Delta) - L(N, \Delta)$ conditional on Θ is

$$\begin{aligned} Q - (1 - Q)((c_1 + c_2)\sqrt{(v^p)}\phi(m^p/\sqrt{(v^p)}) \\ + m^p((c_1 + c_2)\Phi(m^p/\sqrt{(v^p)}) - c_2)) \end{aligned} \quad (3)$$

where $\Phi(z) = \int_{-\infty}^z \phi(u)du$. We declare a change if the posterior average of the function in (3) is negative. A closer look at the mathematics shows that small values of τ^2/σ^2 and/or large values of p makes it harder to reject the null hypotheses.

D. Algorithm for Data Streams

Commonly used approaches to compute δ_{it} and s_{it}^2 when monitoring data streams include **sliding windows** and **exponential smoothing**. Here, we use the former where for a window size n , $\delta_{it} = (y_{it} - \bar{y}_i)$ with \bar{y}_i being the mean and s_{it}^2 the sample variance computed using observations in the window $[t - n : t - 1]$. We choose $n = 30$ days.

The **hbmix** algorithm is as follows:

- Select window size n , initialize vector $\text{StTime}=\mathbf{0}$.
- At time t , compute δ_{it}, s_{it}^2 for all i such that $t - \text{StTime}[i] > n$.
- Declare changes based on posterior of the expression defined in (3).
- Reset $\text{StTime}[j] = t$ for all streams j which had a change at time t .

E. Experiments and Results

We compare **hbmix** with a simple threshold based algorithm which declares a change if $|\delta_{it}/s_{it}| > M$ and which otherwise follows the same procedure as **hbmix**. Our experiment compares the two methods based on performance at a single time point (the last one in the simulated streams). We simulate K streams at $n + 1$ time points introducing “shocks” only at the last time point and compare the false positive and false negative rates. Since the difference in the false positives and negatives is not symmetric, we tweak the value of M so that the false negative rate matches the one obtained by using **hbmix**.

The experiment is as follows:

- 1) Simulate (μ_1, \dots, μ_K) such that μ_i 's are iid $N(\mu_0, \sigma_0^2)$. We set $\mu_0 = 0, \sigma_0^2 = 1$.
- 2) Simulate (s_1^2, \dots, s_K^2) from $\text{inv-gamma}(\text{scale} = .5, \text{shape} = 2)$. This distribution is quite dispersed with 2.5%, 50%, 97.5% being .09,.30, 2.06 respectively.
- 3) For the i^{th} stream, simulate $n + 1$ observations from $N(\mu_i, s_i^2), i = 1, 2, \dots, K$.
- 4) Add 100 “shocks” to randomly selected streams from $\text{uniform}(-A, A)$ distribution at time $n + 1$ (we use $A = 15$).

TABLE I
COMPARING **THRESHOLD** AND **HBMIX**

K	Total shocks	false neg rate(%)	M	false pos(%) (hbmix)	false pos(%) (threshold)
500	100	11.4	2.8	7.9	12.9
1000	100	12.0	3.1	9.4	15.9
2000	100	14.0	3.3	13.4	21.0
5000	100	14.5	3.6	19.4	30.2

- 5) Change detection is done at $n + 1$ using the first n observations, $n = 10$ for this experiment.
- 6) The threshold M is tweaked to match the false negative rate of **hbmix**.
- 7) Steps 1-6 are repeated r times ($r = 100$ in this case).

Results are reported in Table I.

The simulation clearly shows the usefulness of **hbmix** compared to the threshold-based algorithm especially when the number of events being monitored is large. In principle, **hbmix** could be used to track any combination of features. So far, we have used it to track final actionable class by hour of day and final actionable class by the state where the call originated.

F. Derived Features: Dialog Success

Dialog Success from Unlabeled Data: The features described in Section IV are all directly extracted from the system logs. In this section we will describe experiments in deriving features related to some measure of “dialog success.”

The logs themselves yield no direct clues as to whether or not a dialog was successful. Definitions of dialog success tend to rely on features such as task completion rates, time spent, and user satisfaction [13]. Task completion is perhaps the most basic of these measures, but here we do not have certain knowledge as to the task the caller is attempting or whether reaching a particular routing destination constitutes success. Nor do we have user satisfaction data. Nonetheless, we would like to be able to display some indication of dialog success derived fully automatically from the log data alone.

Our ultimate aim is to develop such measures of success and *validate* them using various human sources of judgement. In this section, we describe our first experiments in this area.

Target Success Measure: A small amount of the turn audio data is labeled with transcription and call types by hand, as part of the ongoing collection of training material for the speech recognition and classification modules of the application.

We currently use these turn labels to generate a **turn label based (tlb)** success value for each dialog. (Recall that we earlier defined the **final actionable call type** to be the last call type the classifier obtained before routing or hangup which does not fall into a predetermined set of vague or uninformative call types, or simply the last if none such exist.) If the final actionable call type is among the human-assigned labels for the dialog, the dialog is successful, otherwise it is a failure.

Predicting Dialog Success: We have trained a number of classifiers to predict dialog success for unlabeled dialogs, with

the **tlb** success measure as the target, using various different combinations of features in the unlabeled dialogs.

We use the BoosTexter text classifier [7], formulating our essentially binary classification problem as a two-class problem with classes Y (success) and N (failure).

The training data consists of a set of example dialogs each accompanied by a **tlb** label $\ell \in \{Y, N\}$ derived as described above. Each dialog is represented as a set of textual or real-valued features selected according to the various experimental cases described below. BoosTexter constructs a simple categorization rule, a **decision tree stub**, for each term (word or ngram) in each text feature in the training data. Each such rule makes a prediction, with a confidence score, about each possible output label of a test dialog based solely on the presence or absence of that term in the dialog’s feature set. (Rules for continuous-valued features test for values falling above or below some threshold). We can say that the rules generate real-valued **weak hypotheses** of the form $h(x, \ell) = c_{0\ell}$ (if term is not in input x) or $c_{1\ell}$ (if term is in input x) for each example x and label ℓ .

The training phase is based on the idea of maintaining a distribution D_t of weights over the examples and labels in the training data to force the training process to concentrate on the examples and labels that have proved hardest to classify on the previous round. Initially this distribution is uniform. At each round of training t , the weak hypothesis h_t is chosen which minimizes the error on the (weighted) training set. The parameter α_t which will be used to weight the contribution of h_t at classification time is computed based on its error on the training set. Then the distribution is updated so that D_{t+1} gives more weight to the examples and labels which are most seriously misclassified with D_t (“boosts” them).

When used during classification, the score for a given label ℓ for dialog x is $f(x, \ell) = \sum_{t=1}^T \alpha_t h_t(x, \ell)$ where h_t is the weak hypothesis chosen on round t of training, and α_t is the weight assigned to it. The label with the best score is taken to be the prediction for x .

Experimental setup: We have tested the following combinations of turn and dialog features directly extracted from the logs:

- C (dialog contextual features): final actionable call type; routing destination.
- N (turn features): sequence of turn call types; sequence of prompt ids. These are treated as gappy (sparse) n-gram features up to size 3, that is, up to three consecutive tokens including possible wild cards matching any token.
- T (time features): number of turns; dialog duration; average turn duration; duration of first turn.

The n-gram turn features are interesting because they concisely designate sets of dialog trajectories, taking advantage of BoosTexter capabilities intended for natural language sequences. For instance, the gappy n-gram `p123 * p56`, a possible value for the turns prompt sequence feature, designates all dialogs containing a sequence of three turns where the first has prompt `p123`, the second can have any prompt, and the third has prompt `p56`. Finding which trajectory fragments are associated with successful and unsuccessful dialogs should provide useful insight for the designers of the dialog structure.

TABLE II
CONFUSION MATRICES FOR PREDICTIONS OF DIALOG SUCCESS

		C		T		N		C+T+N	
		N	Y	N	Y	N	Y	N	Y
truth	N	72	128	0	200	48	152	79	121
	Y	36	764	7	793	28	772	36	764

TABLE III
ERROR RATES AND κ VALUES FOR PREDICTIONS OF DIALOG SUCCESS

Classifier	Error (%)	κ
C	16.4%	0.38
T	20.7%	-0.01
N	18.0%	0.27
C+T+N	15.7%	0.42

Training for each of these feature sets used 300 rounds of boosting on 5,217 dialogs labeled with $\ell \in \{Y, N\}$ from a VoiceTone customer care application. Testing used 1,000 held-out dialogs from the same dataset.

Results: Given this setup, the dialog success measure can be predicted from the log features. Confusion matrices are shown in Table II. Error rates and κ values are given in Table III. Results are given for classifiers trained on the various feature sets plus a combination, compared with the **tlb** dialog success measure as “truth”. The combination of basic features, turn features, and time features performs best of those tested. Error rates for this case (disagreement/total) are around 16%. Only the C+T+N classifier has $\kappa \geq 0.4$, the level required to claim “moderate” agreement on κ according to [14].

This level of performance may be useful enough to consider adding this measure of dialog success as a feature to be tracked on the Daily News web site. It should be borne in mind, however, that the predicted success rates (around 89%) are all significantly higher than the “true” success rate of 80%.

Moreover, since the meaning of the turn-label based dialog success measure we are using as the target is itself somewhat unclear, we plan to conduct human evaluations of dialog-level success to validate both the **tlb** measure and to develop new log-based measures.

VI. INFORMATION PRESENTATION

The frequency and alert information for all tracked features plus a large number of pre-constructed plots accompanied by XML descriptions are processed fully automatically into an interactive, hierarchically organized web-site. Alerts and carefully selected summary plots and tables provide high-level information and serve as entry points for exploration of a volume of data which is intractably large without such prioritization. Users can investigate alerts by querying the database, conducting their own analyses and drilling down to specific dialogs and audio segments.

A. Web-site Top Level

A typical front page of the web site is shown in Fig. 3.

Daily tables: On the right, some tables give the top few values for features that are nearly always of interest to the end user, with arrows (after the manner of a stock tracking page)

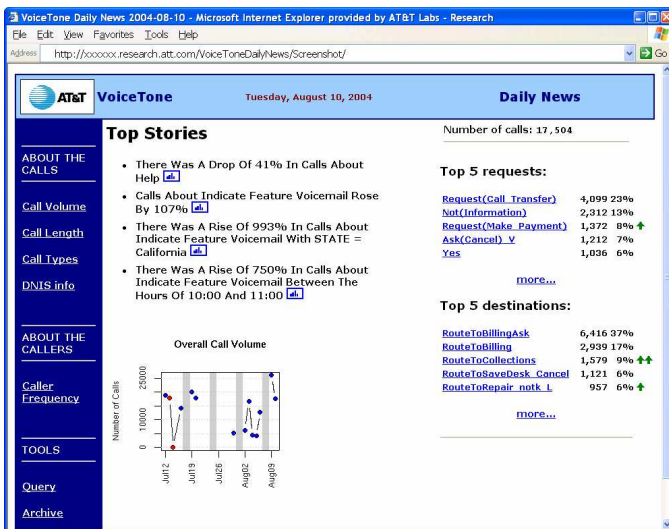


Fig. 3. The front page: a simulated example.

giving a coarse-grained indication of the magnitude of movement for each feature-value pair. Feature-value pairs which exceed the alerting threshold according to equation 1 (e.g., **RouteToCollections** in Fig. 3) are accompanied by double arrows; those which exceed the minor alerting threshold have a single arrow (e.g., **Request(Make.Payment)**).

Each line in these tables is a link to a page showing a plot of the feature-value pair's frequency and proportional frequency movements over the period of the moving window of trend tracking. These Time Series plots (see Fig. 4(a)) plainly show overall trends over the past few weeks, weekly periodic trends, and outliers. This plot type is also used prominently on the front page for the *Overall Call Volume* plot, which displays the number of calls into the system over the most recent time period. The plot is visually broken into weekly segments by light gray vertical bars on the weekends, which allow the weekly trends to be seen more readily.

Detailed statistics: On the left, there are links to pages of plots and tables detailing information about the calls and callers over different periods of time.

For instance, under the "Call Volume" link we show a map which color-codes states based on the volume of calls originating there. A map (see Fig. 4(d)) is a convenient way of seeing if there are spatial effects in the data. We plan to extend this to plot spatial units more granular than the state level. Our maps divide the volume levels into four bins and use a monochromatic color sequence (off-white to red) to designate the volumes in different areas.

On the same page we show a histogram of the number of calls by time of day. These plots allow the user to see modes, skewness and outliers in a convenient form. The example in Fig. 4(c) clearly shows a dip in volume around lunch-time.

Under the "Call Length" link on the left-hand panel we show a boxplot of call length in seconds for the top few final actionable class values (Fig. 4(b)). Boxplots show interactions between two variables in a concise manner. Usually, one is a categorical independent variable and the other a real valued response variable. The box in the middle represents

the interquartile range, with a line drawn at the median, and whiskers extending out to the minimum and maximum of the data.

News headlines: The middle section contains the day's news headlines. Features generating an alert are here brought to the user's attention. Links on each headline lead to a trend plot page, as for the right-hand column table links. In these plots, the sequence of actual values over time is shown along with bounds indicating the currently expected range of volume for the feature in question.

Newsworthy features generate front-page headlines in simple natural language using XSLT templates. Input from the alerting module takes the form of trend elements. For example, the last headline in Fig. 3 is generated from the trend element

```
<trend variable='Calls' change='750'>
  <feature name='FinalActionableClass'
    value='IndicateFeatureVoicemail'
    number='1' />
  <feature name='HourOfDay'
    value='10'
    number='2' /></trend>
```

which represents a bivariate alert on calls with final actionable class **IndicateFeatureVoicemail** and hour of day **10**.

Heavy use has been made of the **mode** feature of XSLT to allow a given piece of input data to be realized differently depending on the linguistic context (following [15], [16]). We also use a simple randomizing mechanism to produce lexical and structural variation.

For example, there is a template which matches trend nodes with positive change attributes in mode **MovementV**:

```
<xsl:template mode='MovementV'
  match='trend[number(@change) > 0]'>
  ...
```

In the body of this template, a random number is generated to control the choice among the possible outputs *rose*, *went up*, and *increased*. A corresponding template in the same mode matches trends with negative changes, generating *fell*, *decreased*, or *went down*.

These **MovementV** templates are used to realize the change attribute of a trend in the context of sentences with structure

```
VariableDescriptionNP MovementV 'by'
ValueMagnitude
```

while in sentences with the alternative structure

```
'there was a' MovementNP
'of' ValueMagnitude 'in'
VariableDescriptionNP
```

the same change attribute will instead be matched in mode **MovementNP** to generate a noun phrase like *a drop*, *a fall* for negative change values, or *a rise*, *an increase* for positive values. The choice between the two sentence-level structures is made using the same randomization idiom.

The trend example above generates "calls about *Indicate Feature Voicemail* rose by 750% between the hours of 10:00 and 11:00," "the number of calls about *Indicate Feature Voicemail* increased by 750% between 10pm and 11am," etc., as well as "there was a rise of 750% in calls about *Indicate*

Feature Voicemail between the hours of 10:00 and 11:00” which appears in Fig. 3.

The use of XSLT templates allows modular expansion of the coverage as different features or trend variables are introduced. There are VariableDescriptionNP templates that match trends with variable=‘Callers’ or variable=‘Calls’. These templates in turn call FeatureDescriptionPP templates, which by default (with no match restriction) merely output *feature=value*. When a new feature is added — for instance the originating state of the call — appropriate behavior can be produced by simply adding one or more new FeatureDescriptionNP templates that match the new feature nodes better than the default templates do, e.g.:

```
<xsl:template mode='FeatureDescriptionPP'
  match='feature[@name='State']'>
  <xsl:text> from </xsl:text>
  <xsl:value-of select='@name' />
</xsl:template>
```

B. Drill-Down

A link from a headline or a daily table entry allows the user to query the database to retrieve more detailed information about the dialogs behind the headlines. Following these links brings up a form to query the database, pre-populated to select dialogs from the current day and the feature-value pair that is currently being investigated. All query features can be varied in the form, for instance to add other values for the same feature in order to make comparisons.

There are currently four display options for the selected dialogs from which the user can further drill down.

Turn-level display and audio access: The default display is a table which shows a line for each dialog, corresponding to the dialog’s record in the database. The drill-down facility is further extended to the turn level: a dialog may be selected to show a page with the sequence of turns, with prompt, recognition output and call type for each turn. There is also a link to the comprehensive system log of that dialog, which shows prompts, responses, recognition, understanding, system state, etc. Specific calls can be examined for clues as to the cause of an alert. This log itself has links to the actual audio files of the recorded utterances and allows listening to the complete dialog, e.g. to probe speech recognition issues.

Comparison plots: The user can compare the distribution of the attributes from the dialogs returned by the query with a baseline set of dialogs. These plots display histograms of each attribute (e.g. number of turns) side by side with plots of the same attributes for a different set of dialogs. Using these plots, one can do a simple investigation by comparing the distributions for dialogs associated with an alert to distributions for a baseline set of dialogs. Currently we are using a single baseline set for all queries; in future work we aim to tailor the baseline dialogs to be the most sensible comparison set for a given query.

SeeCalls: The selected dialogs may be sent to SeeCalls, a highly interactive visualization and data exploration tool designed for transaction data. For VoiceTone data, we display

a set of dialogs as a series of vertical spikes positioned along a time axis, with height proportional to dialog duration (see Fig. 4(e)). This display is linked to a set of barplots corresponding to the attributes in the dialog database, e.g., RoutingDestination or number of turns, so the spikes are usually colored according to one of those attributes. In addition, the main plot is often manipulated and queried by zooming in on a region of the data to view the plot in more detail or to extract database records.

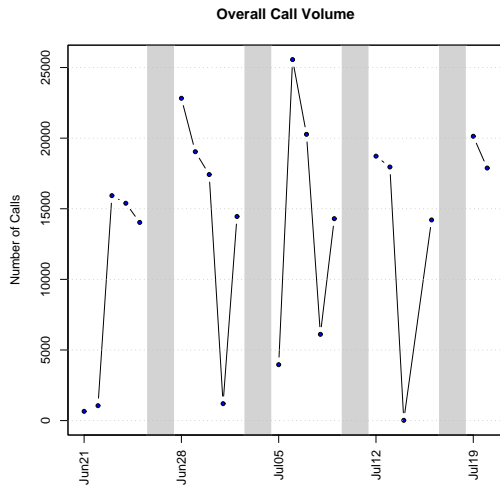
Dialog trajectories: We use technology described in [17] to show a directed graph representation of the call-flow of the set of dialogs selected by the query.

VII. SUMMARY

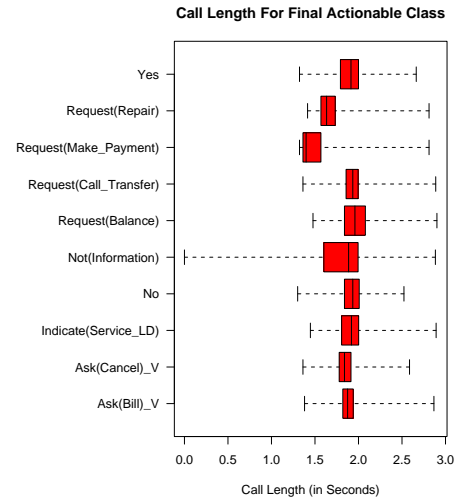
Applying state of the art statistical analysis and visualization techniques to the dynamic behavior of spoken dialog systems offers great benefits both for system evaluation and diagnosis, and for domain-oriented data mining. In this paper we have described the VoiceTone Daily News system for generating newspaper-style reports fully automatically on a daily basis from spoken dialog system logs. The Daily News uses a combination of natural language alerting, tables, and carefully chosen graphics to provide high-level access points to data of interest. A comprehensive drill-down capacity allows the user to select and view parts of the data in various ways, either as aggregates through graphical visualization, or down to the granularity of individual dialogs and turns.

REFERENCES

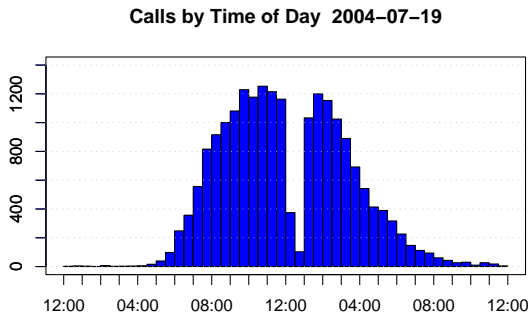
- [1] AT&T, See <http://www.att.com>.
- [2] Nuance Communications, Inc, See <http://www.nuance.com>.
- [3] Scansoft, Inc, See <http://www.scansoft.com>.
- [4] Anonymous, “Mining Customer Care Dialogs for ‘Daily News,’” in *ICSLP*, Jeju, Korea, 2004, to appear.
- [5] AT&T, VoiceTone: <http://www.att.com/products/productdetails.jsp?productId=voicetone>.
- [6] A. Ljolje, D. Hindle, M. Riley, and R. Sproat, “The AT&T LVCSR-2000 System,” in *Speech Transcription Workshop*, Univ. of Maryland, May 2000.
- [7] R. E. Schapire and Y. Singer, “BoosTexter: A boosting-based system for text categorization,” *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.
- [8] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, “The AT&T Next-Gen TTS System,” in *JOINT MEETING OF ASA, EAA, AND DAGA*, Berlin, Germany, March 1999.
- [9] W. W. W. Consortium, “XSL Transformations (XSLT),” W3C Recommendation. See <http://www.w3.org/TR/xslt>.
- [10] R. Greer, “Daytona and the Fourth Generation Language Cymbal,” in *Database Conference*. Philadelphia, PA: ACM SIGMOD, June 1999, See also <http://www.research.att.com/projects/daytona>.
- [11] G. E. Box, *Time series analysis : forecasting and control*. Holden-Day, 1970.
- [12] D. Agarwal, “Detecting Change in Multiple Data Streams using Two-component Gaussian Mixture,” in *NIPS 04*, 2004, under review.
- [13] M. Walker, D. Litman, C. Kamm, and A. Abella, “Evaluating Spoken Dialogue Agents with PARADISE: Two Case Studies,” *Computer Speech and Language*, vol. 12, no. 3, 1998.
- [14] J. Landis and G. Koch, “The measurement of observer agreement for categorical data,” *Biometrics*, vol. 45, pp. 255–268, 1977.
- [15] A. Cawsey, “Presenting tailored resource descriptions: will XSLT do the job?” in *9th International Conference on the World Wide Web*, May 2000.
- [16] G. Wilcock, “Pipelines, Templates, and Transformations: XML for Natural Language Generation,” in *1st NLP and XML Workshop*, N. Nomura and C. Nakabasami, Eds., Tokyo, Japan, November 2001.
- [17] A. Abella, J. Wright, and A. Gorin, “Dialog Trajectory Analysis,” in *ICASSP*. Montreal, Quebec, Canada: IEEE, 2004.



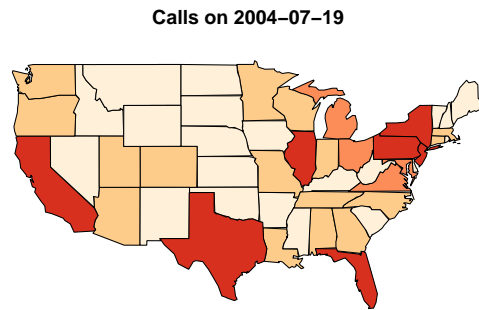
(a) Time series plot of the number of calls per day during a month. The gray vertical bars mark the weekends, and make it easier to see weekly trends.



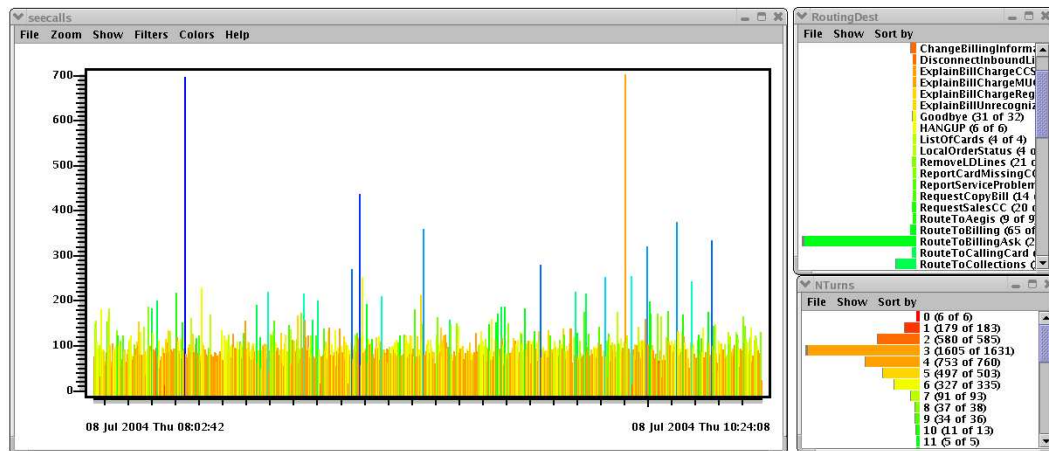
(b) Boxplots displaying the distribution of call length conditioning on Final Actionable Class. The box in the middle represents the interquartile range, with a line drawn at the median, and whiskers extending out to the minimum and maximum of the data.



(c) Histogram of the number of calls by time of day. Note the dip in volume around lunch-time.



(d) Map of the number of calls by state.



(e) SeeCalls display of dialog duration in a two-hour span. The two barplots of dialog features at the right can be linked to the main plot; many other interactions are possible.

Fig. 4. The five main plot categories