



# Online Call Quality Monitoring for Automating Agent-Based Call Centers

Woosung Kim

Convergys Innovation Center  
201 E. 4th Street, Cincinnati, OH 45202, USA  
woosung.kim@convergys.com

## Abstract

One of the challenges in automating a call center is the trade-off between customer satisfaction and the cost of human agents: i.e., most callers prefer human agents to automated systems, but adding human agents substantially increases call center operating costs. One possible compromise is to let callers use automation at the beginning of the call and bring in a human agent if they have problems. The key problem here is, obviously, how to detect the problematic calls *promptly* before it is too late. This paper proposes a novel method for monitoring call quality, aiming to salvage callers having problems with automation by bringing in a human agent in a timely manner. We propose to use finite state machines to automatically label call data for training and use the log likelihood ratio for monitoring calls to detect bad calls. We demonstrate, by experiments, that it is possible to detect bad calls before callers give up the call, which increases customer satisfaction and minimizes costs.

**Index Terms:** call quality monitoring, call center automation, interactive voice response, spoken dialogue systems.

## 1. Introduction

With the advances in computer telephony integration and automatic speech recognition (ASR) technologies, more call centers are being automated. Consequently, today, most call centers are using such automation at least in part. For example, if you make a call to a call center, chances are either a call routing system, which has the natural language understanding (NLU) capability, gets the call asking you a “How may I help you?”-style open prompt or a menu-driven interactive voice response (IVR) system takes the call asking you a “Please select the item from the menu.”-style prompt. Needless to say, the biggest benefit from such automation is cost reduction, especially cutting down agent call handling time since agent costs make up a significant portion of call center operation costs.

Recent advances in ASR have made it possible for an IVR to recognize caller’s speech, allowing callers to directly *speak* the menu item they want as well as to press the touch-tone key. Due to the limitations of the current ASR technology, however, these *speech-enabled* IVRs are not perfect, often misrecognizing the caller’s speech, which causes callers to become confused, frustrated, or even angry. Accordingly, these recognition errors decrease customer satisfaction and further make callers reluctant to use such automated IVRs. Indeed, many callers try to bypass the IVR by saying “customer service” or “human representative” from the beginning of the call without giving a chance to the IVR—even if they can be easily served. Without a doubt, such behaviors are undesirable as they will increase the cost of human agents to serve the callers. From the call center operator’s viewpoint, therefore, it is critical to solve the trade-off between customer satisfaction and costs.

One simple solution to this problem is to let callers use the IVR first and bring in a human agent only when the callers need help for some reason. For example, the IVR may keep misrecognizing caller’s speech, or the caller may be locked in the loop of the call flow. Such situations may be easily resolved by a human agent who is familiar with the IVR and the call flow. The natural questions here are when is best to bring in a human agent and how to decide the right moment to bring in a human agent. Our answer to these questions is *online call monitoring*, the main subject of this paper, that can constantly evaluate and dynamically decide, if and when, to introduce an agent from a remote call center to respond to the caller and complete the call.

## 2. Related work

While there have been a number of studies on measuring the quality of human-computer dialogues or identifying problems in dialogues, little research has been done on online call monitoring. In particular, [1] describes work on automated call quality monitoring, but it classifies dialogues between a caller and a human agent, not an IVR. Also, [2] studies how to evaluate IVRs; but measures IVR’s performance, not call quality, to improve the IVR by re-designing IVR’s call flow. In addition, [3, 4] investigates methods to *predict* problematic caller-IVR dialogues using many features from acoustic/ASR, NLU, and dialogue manager. But the decision to class a call either “good” or “bad” is always made after a fixed number of turns are exchanged, which leaves a major shortcoming: i.e., *when is best to decide* remains unanswered. Similarly, [5] applies the same classifier to simpler features such as prompt types and unigrams of the ASR lattice, but detects problems at the turn level, not at the dialogue level. Finally, [6] describes methods to predict errors of caller-IVR dialogues in an online manner. Though this looks most similar to our work, it rather focuses on adapting the IVR’s dialogue strategy after a bad call is predicted, and yet omits reporting online detection performance. Besides, the prediction is made based solely on ASR confidences without taking into account the recognition result. Finally, it does not generate call quality scores or offer flexibility in choosing bad calls.

Unlike those studies, however, it is indispensable in our work to constantly monitor calls and detect bad calls

- 1) promptly before it’s too late: i.e., before the callers hang up the call or opt out of automation<sup>1</sup>—as we don’t want to lose customers, and
- 2) *dynamically* depending on the situation: e.g., the number of available agents or the lifetime value of the customer, which are the main contributions of this paper and what make our work unique.

<sup>1</sup>Opt-out is a call transferred to a human agent initiated by caller’s explicit request such as “human agent”. It happens when a caller opts to transfer to an agent out of frustration or unwillingness to use the IVR.

---

**IVR:** How may I help you?  
**Caller:** I'd like to transfer \$250 from checking to savings.  
**IVR:** OK, you wanted to transfer \$215 from checking to savings, right?  
**Caller:** No, I said \$250.  
**IVR:** Sorry, I didn't get that. Can you say that again?  
*# The caller gets frustrated and doesn't know what to say.*  
**Caller:** ...  
*# A human agent is introduced.*

---

Figure 1: A sample scenario

### 3. Application: A sample scenario

To illustrate how call monitoring can be efficiently used for automating agent-based call centers, Fig. 1 shows one sample scenario in a banking domain. After a caller-to-IVR dialogue starts as usual, the IVR misrecognizes the dollar amount to be transferred, probably with little confidence. Not fully assured of its recognition result, the IVR asks the caller to confirm if the recognized result is correct. The caller responds with a “No” answer at first because the result is incorrect, but the IVR fails to recognize the answer with a sufficient level of confidence, again. Frustrated when asked to repeat the answer again, the caller is not sure what to say and subsequently hesitates to speak. Soon, the recognizer would regard this as “Time Out”, which is the exactly right moment for a human agent to be introduced, detected and triggered by the call monitoring module. Notice that this agent transfer happens only if the call goes bad. Otherwise, a call would be completed without an agent intervention, which does not require any human agent cost.

In online call monitoring, it is crucial to understand why or when call flows go awry and callers opt out of automation. According to our preliminary analysis, there are many reasons for opt-outs. For example, callers may opt out because 1) the IVR misrecognizes caller’s speech, 2) the IVR does not offer a solution to caller’s needs<sup>2</sup>, or 3) callers simply don’t want to talk to a machine. Obviously, these are not exhaustive, and it is hard to come up with a complete list by heuristics or manual inspections. Moreover, even if we obtain nearly complete reasons from one application, they may be entirely different for another application. Finally, it may not be easy to convert the reasons into a set of rules that can be implemented and used for online call monitoring. Therefore, it is better to *automatically* discover frequent patterns from bad calls and obtain rules for detecting such patterns, which is the main promise of machine learning.

### 4. Online call quality monitoring

The objective in online call monitoring is to detect<sup>3</sup> bad calls before it is too late: i.e., forcing a caller to keep using the IVR even when the call is getting bad will certainly annoy the caller, and this will ultimately result in losing the customer. To this end, it is necessary to estimate (monitor) call quality constantly whenever a caller responds, which may be represented as a quality vs. turn plot. If the quality estimate drops below a certain threshold at any time, the monitoring module signals an alarm to interject a contact center agent. Put another way, all calls are considered to be good by default, and this does not change unless the monitoring module is certain that the call is bad. The next question

<sup>2</sup>The total number of reasons for call may be over several hundreds. In contrast, an IVR supports solutions for only several reasons ( $\leq 10$ ).

<sup>3</sup>Since the goal of call monitoring is to detect bad calls, *monitoring* and *detection* are used interchangeably throughout this paper.

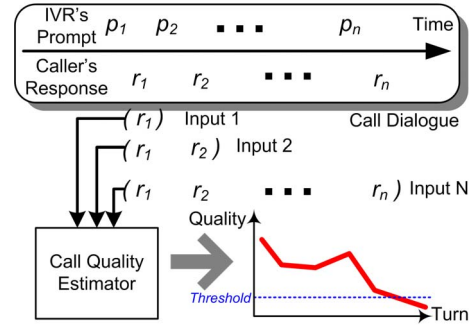


Figure 2: Block diagram of online call monitoring

is, obviously, how to make sure that a call is bad. Our answer to this question is to take a probabilistic pattern classification approach. We first train classification models from training data, and then estimate probabilities of the test input from each of good calls and bad calls. Finally, we base our call quality estimate on those probabilities.

Remember that call quality should be estimated at each turn. E.g., if there are  $N$  caller’s turns (responses) within a call, there will be  $N$  inputs to the call monitoring module. At any turn of the call, since the previous response context (history) is supposedly beneficial for monitoring call quality, we use all previous responses together with the response of the turn, as an input for call monitoring. Therefore, the input is formed as a response sequence up to a certain turn (response sequence prefix or partial input), not a complete response sequence of an entire call. Fig. 2 shows the block diagram with input data.

### 5. Classification models

This section briefly reviews the classification models we use for our experiments. Among many classification models proposed so far, language models (LMs) have been shown useful in text classification [7]. In particular,  $N$ -gram based LMs are built on partial inputs ignoring all context information beyond the previous  $(N - 1)^{th}$  input window. Since the LM-based classifier is believed to be effective in dealing with partial inputs, we have chosen it for our first classification model.

From *class* labeled training data, we can build a class conditional language model  $LM_c$  for each class  $c$  by taking the caller’s response sequence as a word sequence. Assuming a uniform prior, given test input  $x$  consisting of caller responses ( $x = r_1, r_2, \dots, r_m$ ), we can classify it by estimating the likelihood of the sequence from each  $LM_c$ :

$$\hat{c} = \arg \max_{c \in C} P(x|LM_c) \quad (1)$$

where  $C = \{good, bad\}$  in our case.

Our second choice for the classification model is the state-of-the-art BoosTexter [8] classifier based on adaptive boosting. It has been shown effective in many pattern classification problems including spoken language understanding [9]. Boosting first builds weak base classifiers and then iteratively combines them to construct a stronger classifier. That is, a weak classifier is trained to minimize the training error at each iteration. After training, the confidence of a class  $c$  may be estimated by:

$$P(c|x) = \frac{1}{1 + \exp(-2 \cdot \sum_{t=1}^T \alpha_t h_t(x))} \quad (2)$$

where  $h_t(x)$  is the base classifier at iteration  $t$ , and  $\alpha_t$  is its weight as previously used in [10].

## 6. Database

Our database consists of call logs from a call center customer service application on a mailing domain. All caller-IVR interactions such as IVR prompts, ASR transcriptions, meanings, and confidence scores are logged until the call is transferred or hung up. Out of these logs, we use only caller’s *meaning* responses as a feature for classification/detection. In total, there are 38 meaning types including specials such as “NoMatch” (the recognizer’s confidence is too low), “TimeOut” (no input for a certain period of time), and “⟨unk⟩” for all other unknown inputs. After removing noisy calls such as too long (> 20) or short (< 3) turn exchanges, we have about 10K calls which are then divided into training (8K), development (1K), and test (1K) sets.

### 6.1. Automatic data labeling

As there is no indication whether a call is *good* or *bad* in our data, we need to label each call as good or bad prior to building models. Rather than manually labeling the data which requires considerable amounts of time and effort, we propose automatic labeling by using finite state machines (FSMs). An IVR call flow specification which describes IVR’s behaviors has been already made available, and this, in fact, is an FSM. For FSM work, we use the publicly available AT&T FSM toolkit [11].

Once the call flow FSM is available, we can check if a call is valid (or acceptable) according to the call flow: i.e., a call should be accepted by the call flow FSM if the call is valid (good) and rejected otherwise (bad)—by the “fsmcompose” operation in the FSM toolkit. This automatic labeling results in about 55% of good calls and 45% of bad calls. We have excluded the possibility of using FSMs for testing, as it is inappropriate for online monitoring which should be able to handle partial inputs and generate call quality estimates.

### 6.2. Data analysis

Some readers may wonder how bad calls are possible if all calls come from the IVR which behaves as defined in the call flow. In other words, the IVR is supposed to behave exactly as defined in the call flow, and all calls are logged from the same IVR. Thus, all calls should be accepted by the call flow FSM—which would be true, ideally. There are, though, many calls that cannot be accepted for some reasons including:

- **Hang-Up:** According to the call flow FSM, there are final states in which a call may be safely ended or transferred. Our observation, however, reveals that many calls were actually hung up in the middle without reaching the final states. Such calls have not been appropriately completed and thus would be rejected when *composed* with the call flow FSM.
- **Opt-Out:** Technically, opt-out calls should be accepted as they are transferred to a human agent, which is already defined in the call flow. Remember, though, that our goal is to reduce the agent call handling time and opt-outs; therefore, we have, intentionally, made opt-out calls rejected.
- **Others:** A call may be rejected (or we have made it rejected) for many other reasons. E.g., if a caller says many repeated “Yes/No” confirmations<sup>4</sup>, the call should be rejected as these repeated confirmations are indicative of unsuccessful calls. Besides, the call flow includes “failure” states in case when the IVR does not work correctly (e.g., network problems), and if a call falls into such states, it is also rejected.

<sup>4</sup>Remember that confirmations happen only when the recognizer is not sure about the recognition result.

## 7. Experimental results

### 7.1. Off-line classification

We begin our experiments with checking off-line call classification performance. That is, we build caller’s response sequence from the complete call log of each call, from the beginning to the end, and then extract response 5-grams from the sequence as features. We next train the classifiers using the training data, and then test the classifiers. Table 1 shows the result with the accuracy, (*macro-averaged*) precision, recall, and F-score. Keep in mind that this result is based on complete call logs—unlike online detection—and we know that all calls are *ended*. Hence, it should serve as an upper bound for online call detection.

Table 1: *Offline classification performance*

Model	Accuracy	Precision	Recall	F-score
LM-based	.927	.927	.927	.927
BoosTexter	.942	.944	.940	.942

### 7.2. Online detection

Before moving on to online detection experiments, there are some issues to consider. First, we ask ourselves whether it is better to train classification models in an online manner: i.e., we chop up each call into turns and then feed the turn sequence prefixes (up to each turn) to the classifier for training—as is done in testing (Fig. 2). This is necessary to estimate call quality at each turn for online detection. Although this seems plausible, we have decided not to try this, and we keep using the same models as in off-line classification, as 1) there will be too much data if we do, and 2) not all turns (in a bad call) necessarily constitute a bad call. In other words, a call may go bad because of some bad turns, not because of all turns even in the bad call. Second, we have to define how to estimate call quality from partial inputs (response sequence prefixes). Since we have only two models, each from good calls and bad calls, we can compute the log likelihood ratio (LLR) of the two models. E.g., the LLR of the LM-based model<sup>5</sup> can be computed as follows:

$$\text{LLR} = \log \frac{P(x|LM_{good})}{P(x|LM_{bad})}. \quad (3)$$

To see if the LLR is effective for differentiating bad calls from good calls, we proceed to measure the average LLR per turn: i.e., we compute the LLR for response sequence prefixes up to each turn, and average the LLRs by turn. As Fig. 3 shows, it turns out that, for bad calls, the LLR becomes smaller as more turns are exchanged. This plot implies that bad calls may be easily detected by checking the LLR. Interestingly, though, there is no significant quality change for good calls, meaning there is no strong clue for good calls. Satisfied with LLR’s effectiveness to detect bad calls, we use the LLR for measuring call quality.

It still remains undecided what LLR threshold value, which controls when to bring in an agent, should be used. Not only can it be decided empirically as frequently used in optimization, but it may also be decided *dynamically*, e.g., depending on the number of available agents. For our experiments, we decide it empirically from the development set. Fig. 4 shows the accuracy curve for different LLR threshold values along with the off-line classification accuracy (upper bound) for comparison. The curve reveals a modest difference from the upper bound, the curve reveals the best-performing LLR threshold at around  $-1.1$ .

<sup>5</sup>BoosTexter’s LLR is similarly derived from (2) using Bayes’ rule.

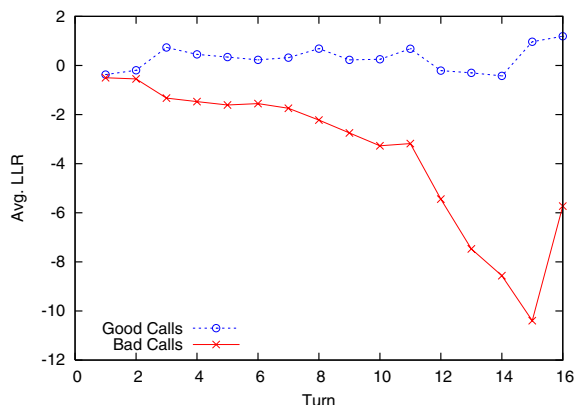


Figure 3: Avg. LLR per turn of good & bad calls

Finally, we use the threshold value ( $-1.1$ ) for online detection against the test set: i.e., if the LLR estimate drops below the threshold, we regard the call as bad. As a result, we have obtained the best accuracy of 83.0% and 73.3% from the LM-based model and BoosTexter, respectively, as in Table 2. Contrary to off-line classification, observe that the LM-based model outperforms BoosTexter. We attribute this to the LM’s effectiveness in dealing with partial inputs. We also notice that there is a difference in accuracy (about 10% absolute in the LM-based model) between off-line classification and online detection; nevertheless, 83% of accuracy in online detection is beyond our expectation considering that the decision is made—regardless whether the call is ended or not—based only on partial inputs. This is notably encouraging as we are able to detect bad calls early. More importantly, it enables us to bring in a human agent to help the callers before they give up the call.

Table 2: Online detection performance

Model	Accuracy	Precision	Recall	F-score
LM-based	.830	.831	.834	.830
BoosTexter	.733	.797	.708	.702

## 8. Conclusions

This paper investigates methods on online call quality monitoring for automating call centers. The main contributions of our study are highlighted as follows. First, we have proposed using FSMs to automatically label call data for training. Next, we have proposed using the LLR of the good calls model vs. the bad calls model to estimate call quality for online call monitoring. In essence, we have established a foundation for online call monitoring and demonstrated its possibility. Our experiments have shown that online call monitoring performs closely to off-line classification performance which is the upper bound. This is remarkable considering that our work is based only on simple caller response sequence prefixes and thus allows early detection of calls that will not successfully complete. By incorporating the LLR threshold value, which may be empirically decided, our approach also enables us to select bad calls dynamically depending on the situation. Finally, our work has a lot of potential—by promptly introducing a human agent—such as deterring callers from giving up the call, further increasing customer satisfaction, and minimizing costs. Encouraged by our results, we are planning to use different features (e.g., ASR confidences) as well and to apply unsupervised learning (e.g., outlier detection) without resorting to labeled data.

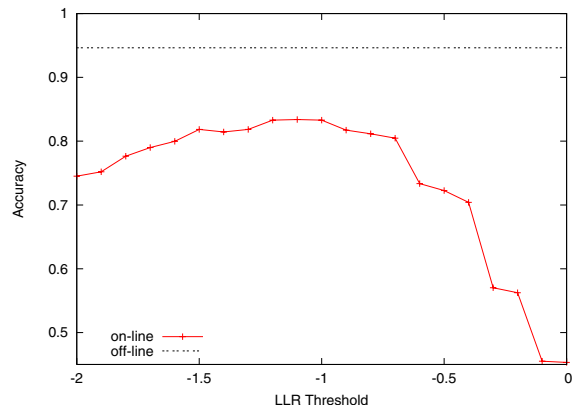


Figure 4: Online detection accuracy vs. LLR threshold

## 9. References

- [1] G. Zweig, O. Siohan, G. Saon, B. Ramabhadran, D. Povey, L. Mangu, and B. Kingsbury, “Automated quality monitoring in the call center with ASR and maximum entropy,” in *Proc. of ICASSP*, vol. 1, Toulouse, France, May 2006, pp. 589–592.
- [2] B. Suhm and P. Peterson, “A data-driven methodology for evaluating and optimizing call center IVRs,” *Int’l Journal of Speech Technology*, vol. 5, no. 1, pp. 23–37, June 2002.
- [3] I. Langkilde, M. Walker, J. Wright, A. Gorin, and D. Litman, “Automatic prediction of problematic human-computer dialogues in ‘How may I help you?’,” in *Proc. of ASRU*, Keystone, CO, USA, Dec. 1999, pp. 369–372.
- [4] M. Walker, I. Langkilde, H. Hastie, J. Wright, and A. Gorin, “Automatically training a problematic dialogue predictor for a spoken dialogue system,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 293–319, 2002.
- [5] A. Bosch, E. Kraemer, and M. Swerts, “Detecting problematic turns in human-machine interactions: Rule-induction versus memory-based learning approaches,” in *Proc. of ACL*, Toulouse, France, July 2001, pp. 499–506.
- [6] D. J. Litman and S. Pan, “Designing and evaluating an adaptive spoken dialogue system,” *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, vol. 12, no. 2-3, pp. 111–137, 2002.
- [7] F. Peng and D. Schuurmans, “Combining Naïve Bayes and N-gram language models for text classification,” in *Proc. of Euro. Conf. on Information Retrieval Research*, Pisa, Italy, April 2003, pp. 335–350.
- [8] 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.
- [9] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Gilbert, “The AT&T spoken language understanding system,” *IEEE Trans. on Audio, Speech and Language Proc.*, vol. 14, no. 1, pp. 213–222, Jan. 2006.
- [10] M. Karahan, D. Hakkani-Tür, G. Riccardi, and G. Tur, “Combining classifiers for spoken language understanding,” in *Proc. of ASRU*, Virgin Islands, USA, Nov. 2003, pp. 589–594.
- [11] M. Mohri, F. Pereira, and M. Riley, “Weighted finite-state transducers in speech recognition,” *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.