SemDial 2011: Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue, pages 158–166.
Los Angeles, California, 21–23 September 2011.

A GLOBAL EXPERIENCE METRIC FOR DIALOG MANAGEMENT IN SPOKEN DIALOG SYSTEMS

Silke Witt
West Interactive
550 S Winchester Blvd
San Jose, CA 95128
switt@west.com

Abstract

This paper presents a metric to automatically track the experience of a caller in a spoken dialog system up to the current moment in time. This metric can be used for two purposes. Firstly, it can be used by the dialog manager to adapt the call flow if the metric reaches a pre-defined threshold. Secondly, it can be used to automatically score the caller experience for each call. This paper will describe the metric itself and how to estimate the parameters for this metric in order to enforce the dialog system to match a set of pre-defined rules as to when to transfer a caller. Additionally, it will be shown that these automatically derived scores correlate well with human ratings and can be used as an automated method to measure overall caller experience in a dialog system. Lastly, data from three live systems utilizing this metric will be presented to show how system performance can be increased by using the metric to aid dialog management.

Index Terms— caller experience metric, dialog management, spoken dialog systems, spoken dialog system evaluation, speech recognition, voice user interface design.

1 Introduction

Generally, in commercial spoken dialog systems, two of the main hurdles in terms of cost efficiency are the need to handcraft every single interaction with the system as well as the requirement that a single system has to handle many different types of users. Such users could be novices or experienced users, cooperative or distracted users, or callers from quiet versus noisy environments etc.

It is due to these hurdles, that no matter how well designed and fine-tuned a spoken dialog system is, there will always be a percentage of callers that will have difficulties interacting with a system and thus will be unsatisfied with the experience. Generally, in dialog systems that automate call center functionality, the balance between automation rate and caller satisfaction is controlled by rules that determine when to transfer a call to a call center agent. The by far most common rule is that after 3 consecutive errors in one dialog state, the caller is being transferred to an agent. However, this approach has the drawback of not taking into account the caller experience up to the dialog state where the errors are happening. This transfer rule also doesn't take into account any other call event type except the specific error type such as a rejection or timeout error. In other words, the transfer decision is based on a single event type as opposed to utilizing multiple features for the decision making.

There have been several previous approaches to measure caller experience and/or to predict problematic calls. Paek, 2001, presents a comprehensive summary of the possibilities and challenges in evaluating spoken dialog systems. Walker et al., 1999 and 2002, describes a method to use the information of the first two to four dialog turns to predict if a caller will experience difficulties, but this method does not apply to every possible dialog state in a system. Evanini et al., 2008, presented a method to calculate the caller experience automatically for an entire call. However, the calculation is derived from application logs after a call is completed. Levin et al., 2006, presented a method to calculate at each turn in a system whether the cost of transferring is less than the cost of keeping the caller in the system.

Likewise the metric presented here is being evaluated at each dialog turn in order to decide whether to continue the current dialog strategy or to switch the dialog strategy. The difference to Evanini is that the metric is calculated at each dialog turn and the transfer decision is based on a threshold around the caller experience rather than the cost.

In summary, this paper will describe the use of a caller experience metric for two main purposes.

I. We will show how such a metric can be used to automatically assign a caller satisfaction score to each call at the end of each call.
II. We will demonstrate the impact on spoken dialog system performance of using such metrics to aid the dialog manager's decision on the next turn in the call.

This paper is organized as follows: Section 2 provides the necessary background on human caller experience ratings, call event types and the relationship between these two. Section 3 presents the core algorithm and parameter estimation method for the caller experience metric (*CEM*). Section 4 discusses the correlation between such automated caller experience scores and human scores. Section 5 presents the results of implementing the *CEM* algorithm in three live spoken dialog systems and lastly, section 6 covers the conclusions.

2 Caller experience ratings and call event types

The purpose of the *CEM* method is to create a metric for the experience of a caller in a spoken dialog system up to the current dialog state. To do so requires accounting for all possible event types that can occur at each dialog state. These event types are:

- **Successful turn:** The system successfully recognized and also confirmed the caller's utterance.
- **Rejection error:** The recognizer could not understand the caller utterance with sufficient confidence and the utterance got rejected.
- **Timeout error:** The system did not detect any caller speech during a predefined time period, typically around 5secs.
- **Disconfirmation:** The caller disconfirmed the recognition result of the system.
- **Agent request:** The caller requested to speak with a call center agent, this can typically be interpreted as a sign that the caller does not want to use the system.

The aim of *CEM* is to create an automated score of the caller experience at the end of a call that can replace a human rating. To do so, requires measuring the correlation between the automated *CEM* score and human ratings. As part of that work, we first generated expert ratings for the same dialog system that we are generating the *CEM* scores for.

2.1 Human caller experience ratings

It is a common practice to evaluate the caller experience that a spoken dialog system provides by having experts score whole call recordings of users interacting with the system in question.
The purpose of the automatic scoring metric presented in section 3 is to replace or at least reduce the need to have human experts score whole call recordings. In order to be able to compare the performance of the automatic scoring method introduced in this paper, we had a human rater score 100 calls for a cable application on a scale of 1 to 5, with 1 being the most positive. The rater was experienced in rating call recordings of this nature and received detailed rating instructions for this particular rating task. The instructions included to count the number of negative call events during the call as well as judging the likelihood that the caller will use the system again, i.e. judging the tone of voice of the caller and how the call is going.

2.2 Typical call event patterns for each rating category

In order to understand the relationship between call event sequences in a call and the rating a human assigned to a given call, Table 1 shows the most common call event sequences for each of the 5 rating types and their associated frequency. These call event sequences and associated frequencies were generated from 23,000 call logs for a cable television company.

Human Rating	Example event sequence	Frequency
5	agent, rejection error, rejection error, agent	0.1%
5	rejection error, succ. turn, rejection error, rejection error	0.3%
4	agent, disconfirm, succ. turn, agent	0.03%
4	disconfirm, agent, nomatch	0.09%
4	disconfirm, disconfirm	0.13%
4	rejection error, succ. turn, rejection error, succ. turn	0.9%
4	succ. turn, rejection error, rejection error, succ. turn	0.3%
3	rejection error, rejection error, succ. turn, succ. turn	1.0%
3	agent, rejection error, successful turn	1.3%
2	Succ. turn, succ. turn, rejection error, succ. turn	0.3%
2	agent, succ. turn, succ. turn	2.4%
2	Timeout error, succ. turn, succ. turn	3%
1	succ. turn, succ. turn.,	22%

Table 1: Example event sequences for calls with negative caller experience

From the event sequences that lead to negative call ratings it can be seen in Table 1 that typically there are at least two negative events such as a rejection error and a disconfirmation. However, two negative events alone do NOT mean that a call will lead to an overall negative caller experience. Rather, the call experience rating depends on the ENTIRE sequence of events throughout a call. For example, a sequence of a rejection error, successful turn, rejection error and again a successful turn can lead to a still acceptable caller experience whereas a rejection error followed by a disconfirmation would lead to a sufficiently negative experience, so that it is advisory to transfer a caller out versus keeping them in the system. In other words, the judgment of a call is not limited to the events in a single dialog state but rather based on the caller experience across several states.
From Table 1 it can also be seen that event patterns for calls with a positive caller experience predominantly have successful turns with only the occasional rejection or timeout error or even only successful turns.

3 Caller Experience Metric (CEM)

Ideally, those callers who are likely to be frustrated and unlikely to be successful in completing their goal are the ones that should be transferred to an agent or presented an alternative modality like touch-tone. On the other hand, callers who might have had occasional recognition or turn-taking errors but otherwise are making progress should continue to be treated as before by the dialog manager.
This can be modeled with what we will call a ‘caller experience metric’, which models the entirety of a caller’s interaction with a system up to the current moment in time as opposed to the interaction at a dialog state level.

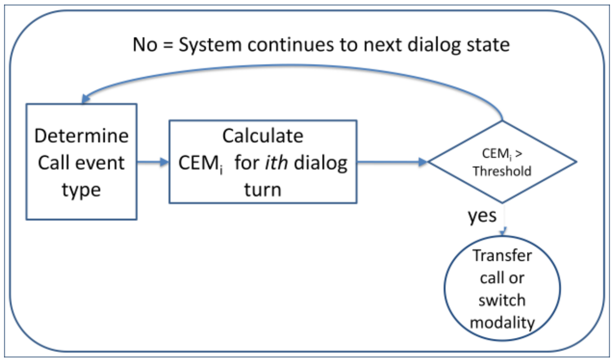
No = System continues to next dialog state
Determine Call event type
Calculate CEM_i for *ith* dialog turn
CEM_i > Threshold
yes
Transfer call or switch modality

Figure 1: CEM architecture describing the CEM calculation at each dialog turn

Figure 1 depicts on overview of this caller experience metric architecture. At every turn in a dialog, the value of this metric is as one of the decision criteria for the dialog manager to decide on the next action. Possible actions are to continue the current mode, to transfer the call or to switch modality, i.e. switch to DTMF, to reduce the prompt readback speed, to change the prompting style and so forth.

3.1 CEM Definition

Let ***S*** be a set of weights for all call event types or setback features that are taken into account for this metric. Such events might be any number of events that describe the caller experience at a given dialog state and are available at runtime.

The set of call events types used in this paper, $s_{k,}$ are:

- Rejection error: $s_{Rej.}$
- Disconfirmation: s_{Dis}
- Timeout error: s_{TO}
- Agent Request: s_A
- Successful Recognition Event: s_{Suc}

Let *d* be a discounting variable to make things further in the past less important. Thus, if a caller had a couple of errors followed by several successful recognition steps, the errors further in the past have less impact.
Then, at each dialog turn *i*, the experience metric gets calculated as

$$CEM(i) = d \cdot CEM(i-1) + s_k(i)$$

Where *CEM(0) = 0* and $s_k(i)$ denotes the weight of caller event type $s_{k,}$ in turn *i.* After calculating *CEM(i)* at each dialog turn, the dialog manager will also check if *CEM(i)* is above a predefined threshold. If the score is above the threshold, the dialog manager will take the predefined action such as transferring the caller out of the application or switching to a different modality such as touch-tone instead of continuing the call in its current mode.

3.2 CEM Parameter Estimation

This section will present a method to estimate the parameter set S as defined in section 3.1.

In order to use this caller experience metric as a dialog management mechanism, one can define a number of rules that describe for which kind of event sequences a call should stay in the application or current modality and for which kind of event sequences a call should be transferred or get some other special treatment. This step is important in a commercial deployment, because clients tend to want to define under which circumstances a caller will be transferred. In other words, this method presented here allows to predefine the system behavior BEFORE a system goes into production (and no statistics on caller behavior are available) and it allows clients (for whom the system has been built) to define the event sequences when callers should be kept in a system and when transferred out.

Based on frequently observed event patterns as shown in Table 1, let us choose six example conditions, where three conditions represent negative event sequences after which a call should pass the threshold. Let's also assume three conditions for positive or acceptable event sequences which should yield a *CEM(i)* score just below the threshold, i.e. a call should continue in its current mode. The choice of the latter three equations should be for moderately successful event sequences.
This is so because a call with only successful turns would always be well below the threshold, whereas we are mostly interested in estimating a set of event type weights that will yield a global score just below the threshold for the acceptable sequences and a score above the threshold for negative event sequences.

For the example here, let's assume the following six event sequences:

(1) *CEM*(i) should be above the threshold after 2 Disconfirms
(2) *CEM(i)* should be above threshold after 1 Disconfirm, 1 agent request and 1 Rejection error.
(3) *CEM(i)* should be above threshold after 2 Rejections, 1 successful turn, another rejection and then a 1 timeout.
(4) *CEM(i)* should stay below threshold for: 1 timeout, 1 successful turn, 1 rejection and an agent request.
(5) *CEM(i)* should stay below threshold for: 1 disconfirmation, 1 successful turn, 1 timeout
(6) *CEM(i)* should stay below threshold for 1 successful turn, 1 rejection, 1 timeout, 1 successful turn and 1 timeout.

Note that these sequences are examples only in order to illustrate the process of parameter estimation. These equations need to be separately chosen for each dialog system before it goes into production.

Now, let T denote the decision threshold. Then, the *CEM(i)* score after the completion of these event sequences can be calculated by recursively plugging all events into the CEM formula. Doing this for the 6 example sequences yields the following set of inequalities:

I. $(1+d)s_{Dis} \quad > T$
II. $d^2 s_{Dis} + ds_A + s_{Rej} \quad > T$
III. $(d^4+d^3+d)s_{Rej} + d^3\, s_{Suc,} + s_{TO} \quad > T$
IV. $d^3 s_{TO} + d^2 s_{Suc} + ds_{Rej} + s_A \quad < T$
V. $d^2 s_{Dis} + d\, s_{Suc} + s_{TO} \quad < T$
VI. $(d^4+d)s_{suc} + d^3 s_{Rej} + (d^2+1)s_{TO} \quad < T$

In order to convert these inequalities into a set of equations, let ε be an offset value by which the *CEM* score should be above the threshold in order to meet the transfer condition for the first three event sequences and below the threshold for the last three event sequences. With this, we arrive at the following equation system:

I. $(1+d)s_{Dis} \quad = T + \varepsilon$
II. $d^2 s_{Dis} + ds_A + s_{Rej.} \quad = T + \varepsilon$
III. $(d^4 + d^3 + d)s_{Rej.} + d^3 s_{Suc} + s_{TO} = T + \varepsilon$
IV. $d^3 s_{TO} + d^2 s_{Suc} + ds_{Rej.} + s_A \quad = T - \varepsilon$
V. $d^2 s_{Dis} + d\, s_{Suc} + s_{TO} \quad = T - \varepsilon$
VI. $(d^4+d)s_{suc} + d^3 s_{Rej} + (d^2+1)s_{TO} = T - \varepsilon$

Now, let ***s*** be a vector of the to-be-estimated event type weights, i.e.:

$$\boldsymbol{s} = \begin{bmatrix} s_{Dis} \\ s_{Suc} \\ s_{Rej.} \\ s_A \\ s_{TO} \\ -T \end{bmatrix}$$

And let **ε** be a delta vector to reflect the score after a given event sequence to be below or above the threshold:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \\ -\varepsilon \\ -\varepsilon \\ -\varepsilon \end{bmatrix}$$

Then, the set of six equations can be rewritten as a vector equation:

$$\boldsymbol{F} \times \boldsymbol{s} = \boldsymbol{\varepsilon}$$

Solving this equation system for the set of weights ***s*** leads to:

$$\boldsymbol{s} = \boldsymbol{F}^{-1} \times \boldsymbol{\varepsilon}$$

And with this equation, we now have a simple expression to calculate an estimate of ***s*** for a predefined set of event sequence behaviors.

There are two requirements for this equation to be solvable:

First, the number of chosen call event sequences has to match the number of parameters to be estimated so that ***F*** becomes a square matrix. Secondly, the example call event sequences have to be chosen so that the resulting matrix ***F*** will have full rank and thus is invertible.

Assuming a discount factor $d = 0.9$ and the offset constant $\varepsilon = 0.5$, Table 3 shows the estimated parameter set for the solution of our six example equations above. These resulting parameter values make intuitively sense. For example, disconfirmations, which tend to have quite a negative impact on caller experience, have the highest weight, whereas the weight for an agent request is much smaller, since such an event is caller initiated and doesn't have quite such a bad impact on the caller experience. A successful turn tends to improve the caller experience and this matches the negative weight for s_{Suc}.

Parameter Name	Estimated value
s_{Dis}	1.46
s_A	0.66
s_{Rej}	1.00
s_{Suc}	-0.46
s_{TO}	0.80
T	2.27

Table 2: parameter estimates for the example equation system

It is important to note that the weights listed in Table 2, are only example results. The values of ***s*** depend heavily on the choice of the six constraining equations as well as the default settings for *d* and ε. The algorithm presented here can be seen as a framework to estimate a set of caller event weights that best matches the requirement for a specific system.

3.3 Correlation with Human Scoring

The previous section discussed how to find a set of weight parameters so that predefined set of example call event sequences will result in the desired call handling aka dialog management.

As described by Evanini et al., 2008, the agreement between two raters, in this case between a human and an automatic rater, can be measured with Cohen's κ, see Cohen (1960). This correlation metric factors in the possible agreement between two raters due to chance, *P(e)*. Let *P(a)* be the relative observed agreement between two raters, then κ is defined as:

$$\kappa = \frac{P(a) - P(e)}{1 - P(e)}$$

Since the ratings in this case are on an ordinal scale, we used a linearly weighted κ in to account for the fact that the difference between two adjacent ratings is smaller than the difference between two ratings further apart.
Evanini et al., 2008 conducted an extensive study that showed the correlation in the ratings between human raters and also between the automated metric and a human rater, so we know that

a) ratings by human judges correlate assuming that the raters have been given reliable scoring instructions
b) that it is possible to have automated metrics that can correlate with human ratings.

The purpose of this paper therefore is to validate that the metric proposed in section 3 too can generate automated scores for calls that correlate with human ratings, in addition to being a method to aide dialog management.

In order to correlate the *CEM* score with the human ratings, the *CEM* score (which is a continuous number) was converted to same discrete range of 1 to 5 as the human scores. For the human ratings we used those 100 human ratings as described in section 2. The discount variable *d* has been set to 0.9.

Next, in addition to Cohen's κ, a different way of looking at the correlation between the machine and the human scoring is by measuring which percentage of call received the same rating between the human rater and the machine and how many calls received a rating that differs only by 1 point.

Table 3 shows the κ value and the agreement statistics for different parameter sets. Each row represents one parameter set ***S***, the resulting κ value, the percentage of exact agreement between human and machine, the percentage of agreement differing by 1 and finally the total agreement. Total agreement is defined as the sum of exact agreement and agreement with difference of 1.

The parameter set in row 1 is the parameter set that was found via solving the equation system, see section 3.2. The correlation and agreement is high enough to say that the *CEM* scores correlate with human scores.

Next, the question arose, whether there exist parameter sets that also fulfill the equation set but possibly yield a higher correlation with human raters. To find this out, we manually varied the each of the five parameters while keeping the other four fixed. Row 2 in Table 3 shows the parameter set that yielded the maximum κ value we found by manually varied the weight parameters.

Parameter set #	s_{Rej}	s_{TO}	s_A	s_{Dis}	s_{Suc}	κ	% Agreement between human and machine	%Variance by one between human and machine	% total agreement (up to a difference of 1) between human and machine
1 (estimate from Table 2)	1	0.8	0.66	1.46	-0.5	0.670	64	28	92
2 (max kappa combination)	0.9	1	0.6	1.5	-0.2	0.733	76.6	16.7	93.3
3 (example max overall agreement)	0.9	1	0.4	1.5	-0.2	0.719	70	24.4	94.4

Table 3: Agreement between human and machine ratings for different parameter sets

Lastly, row 3 shows that parameter set found via the manual variations that yielded the maximum overall agreement between machine and humans as opposed to the maximum κ in row 1.

Figure 2 depicts the results of the manual parameter variations in more detail. For each graph, only one of the caller event type weights has been varied, while the rest has been kept constant. As can be seen, the correlation between human and CEM scores at call end is fairly high, independent of the parameter values as long as they are within the valid range. It is interesting to note that the agent requested related weight and especially the rejection related weight have the most influence on the degree of agreement with human scores.

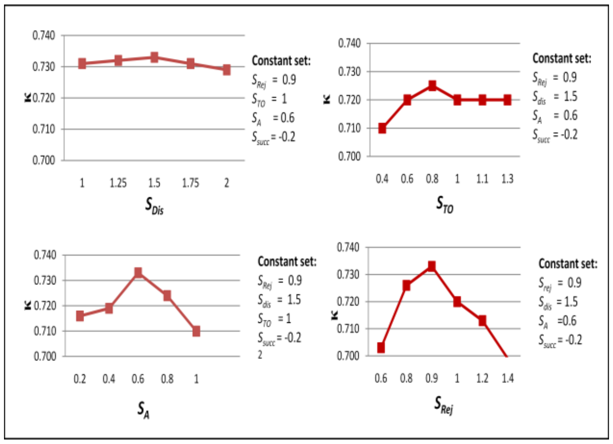

Figure 2: Dependency of Kappa on different caller events

In future work, we will look at optimizing κ via statistical methods in the case that human ratings are available.

The results from Table 3 and Figure 2 show that the correlation between human and CEM scores at call end is high enough, so that the CEM score at call end can be used as an automated rating mechanism in spoken dialog systems.

4 Live system implementation results

The *CEM* scoring was implemented in three live commercial systems. This section will present results for using this metric for both dialog management and measuring caller satisfaction.

4.1 Results for System 1

One of the live systems where the *CEM* scoring described in this paper is currently implemented is a call routing application in the cable television domain.

Generally speaking, high caller satisfaction can be represented by a low average *CEM* score at call end. On the other hand, high automation can be measured by a minimum number of failed calls. Failed calls are defined as calls where the *CEM* score was above a transfer threshold.

Given these definitions, Figure 3 now depicts the relationship between the automation rate (which is the inverse of the %failure calls shown) and different transfer thresholds *T* for this application based on 24036 calls.

With an increasing threshold, callers are kept longer in the application and thus potentially experience more setbacks. This in turn results in an increase of the average *CEM* score at call-end. At the same time the failure rate decreases with an increasing threshold since a higher threshold means calls are likely to be transferred out.

It can be seen that starting around a threshold of 4 and above, the decrease in failure calls as well as the increase in *CEM* levels off and thus a threshold of 4.9 would be a good trade-off value between automation and caller satisfaction (and this is the value that the system currently is using). Note that in this example, the parameters for the *CEM* calculation were based on the 6 equations from section 4.2.

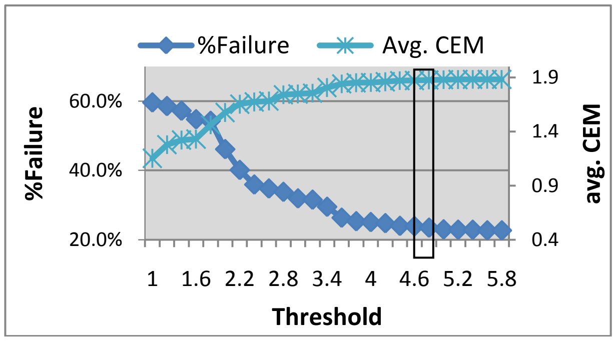

Figure 3: Impact of CEM threshold on caller satisfaction and success based on a live system for a cable company

4.2 Results for System 2

This section shows results for using the *CEM* as a dialog management tool, but this time instead of transferring when the *CEM(i)* score reaches the threshold, the application will instead change the modality from speech to touch-tone. That is, in this case the *CEM* is being used as a metric to gauge the caller experience throughout the call and if the experience is getting bad, the application would switch to touchtone. Using touchtone as a modality makes the interaction more elongated and tedious for a caller, but will at the same time minimize the amount of recognition errors and thus reduce caller frustration.

This is particularly helpful in the case when either a caller has a heavy accent or there is a lot of background noise or side-speech. This approach was chosen, because this system that provides movie show times and ticketing information did not have the option of transferring problem calls to a call center.

Application Configuration	**%Calls ending in Max Error**	**avg. # Error/Call**	**avg CEM**
Baseline (no *CEM*)	3%	2.7	0.90
using *CEM* to switch to touchtone	1%	1.6	0.94

Table 4: Impact of using CEM to switch modality on the overall system performance

Table 4 shows the impact of using the *CEM* score to switch to touchtone. The data is based on reporting statistics for a live system and based on a sample set of over 100,000 calls. The system performance is shown in terms of average number of errors as well as in the % of calls that ended due hitting the max error criterion.

Row 1 shows the baseline performance of the system configured with the standard rule of transferring after three errors. Note that in this case the average *CEM* score was simulated afterwards from log files.

The second row of Table 4 shows the performance after the implementation of *CEM*. Using *CEM* to switch modality if a given threshold was reached, resulted in a 40% decrease in the average number of errors. Overall, the percentage of calls that ended in a max error scenario was reduced by 66%. However, these improvements came at the cost of a slight decrease in the caller experience (since the callers are essentially kept longer in the system). This impact on the caller experience can be seen from the increase in the average *CEM* score at call-end.

4.3 Results for System 3

The third system that has the *CEM* implemented is an application to start,stop or move one's energy service at a home. Just like the previous two systems, this application was coded in a way that allowed changing the event weight values s_i at runtime.

For this application, high automation rates are most important. Therefore, when after an initial release, the automation statistics weren't as high as expected, some of the event weight values were adjusted to essentially keep callers longer in the system. Table 5 shows the fairly large impact of changing the weight values for this commercial application. Again, this data was derived from the reporting statistics of a live system and is based on over 10,000 calls for each system version (before and after).

Application Type	Success rate of Initial Release	Success Rate after CEM Parameter adjustment	Relative Improvement
Stop	57.40%	63.87%	11.27%
Start	5.70%	8.23%	44.39%
Transfer	10.10%	13.39%	32.57%

Table 5: Impact of event weight values changes on overall automation rates

5 Conclusions

This paper presents a metric to measure the caller experience up to the current moment in time during a call. A method to estimate the necessary parameter weights so that the system will behave according to a set of pre-defined rules was also presented.

One of the advantages of this metric is that by pre-defining the rules at system development time, it is possible to account for client business rules as to how a system should behave. Moreover, if the system is programmed so that the weight parameters and threshold are configurable at run-time, the systems behavior can easily be changed imme-

diately. For example, if a call center is experiencing high wait times, one can increase the threshold, thus keeping more callers in automation and thus have less traffic to the call center at the cost of a less good experience for some callers.
It was shown that the score of this automated metric at call end correlates well with human rating. Thus this metric can be used for two reporting purposes: First, to automatically flag problem calls. Secondly, the average of this metric at call end can be used to directly measure caller experience over time.

Moreover, using this metric as a decision criterion for dialog management has been shown to improve the automation in a live system.

Future work will focus on expanding the set of features contributing to the metric and on expanding the range of actions the dialog manager might take when the threshold is being reached.

6 References

Jacob Cohen, "A coefficient of agreement for nominal scores," Educational and Psycological Measurement, vol. 20, no. 1, pp. 37-46, 1960.

K. Evanini, P. Hunter, J. Liscombe, D. Suendermann, K. Dayanidhi, R. Pieraccini. 2008. Caller Experience: A method for evaluating dialog systems and its automatic prediction. Proc IEEE Workshop on Spoken Language Technology (SLT), Columbus, Ohio, USA..

E. Levin, R. Pieraccini. 2006. Value-based optimal decision for dialog systems. Proc IEEE Workshop on Spoken Language Technology (SLT), Aruba.

T. Paek, 2001, Empirical Methods for Evaluating Dialog Systems, Proceedings of the workshop on Evaluation for Language and Dialogue Systems.

T. Paek and E. Horvitz. 2004, Optimizing Automated Call Routing by Integrating Spoken Dialog Models with Queuing Models. Proc. Of HLT-NAAC 2004, pp. 41 – 48.

I. Langkilde, M. Walker, J. Wright, A. Gorin and D. Litman, Automatic Prediction of Problematic Human-Computer Dialogues in How May I Help you?, Proc. Of ASRU 1999.

M. A. Walker, I. Langkilde-Geary, H.W. Hastie, J. Wright, A. Gorin, Automatically Training A Probematic Dialogue Predictor for a Spoken Dialog System, Journal of Artificial Intelligence Research, Vol. 16 (2002), p 293-319, 2002.

M.A. Walker, I. Langkilde, J. Wright, A. Gorin, D. Litman, Learning to Predict Problematic Situations in a Spoken Dialogue System: Experiments with How May I Help You?, NA Meeting of ACL, 2000.