

# Automatic Detection of Miscommunication in Spoken Dialogue Systems

Raveesh Meena    José Lopes    Gabriel Skantze    Joakim Gustafson

KTH Royal Institute of Technology

School of Computer Science and Communication

Stockholm, Sweden

{raveesh,jdlopes,skantze,jkgu}@kth.se

## Abstract

In this paper, we present a data-driven approach for detecting instances of miscommunication in dialogue system interactions. A range of generic features that are both automatically extractable and manually annotated were used to train two models for online detection and one for offline analysis. Online detection could be used to raise the error awareness of the system, whereas offline detection could be used by a system designer to identify potential flaws in the dialogue design. In experimental evaluations on system logs from three different dialogue systems that vary in their dialogue strategy, the proposed models performed substantially better than the majority class baseline models.

## 1 Introduction

Miscommunication is a frequent phenomenon in both human–human and human–machine interactions. However, while human conversational partners are skilled at detecting and resolving problems, state-of-the-art dialogue systems often have problems with this. Various works have been reported on detection of errors in human–machine dialogues. While the common theme among these works is to use error detection for making online adaption of dialogue strategies (e.g., implicit vs. explicit confirmations), they differ in what they model as error. For example, Walker et al. (2000) model dialogue success or failure as error, Bohus & Rudnicky (2002) refers to lack of confidence in understanding user intentions as error, Schmitt et al. (2011) use the

notion of interaction quality in a dialogue as an estimate of errors at arbitrary point in a dialogue, Krahrmer et al. (2001) and Swerts et al. (2000) model misunderstandings on the system’s part as errors.

Awareness about errors in dialogues, however, has relevance not only for making online decisions, but also for dialogue system designers. Access to information about in which states the dialogue fails or runs into trouble could enable system designers to identify potential flaws in the dialogue design. Unfortunately, this type of error analysis is typically done manually, which is laborious and time consuming. Automation of this task has high relevance for dialogue system developers, particularly for interactive voice response (IVR) systems.

In this paper, we present a data-driven approach for detection of miscommunication in dialogue system interactions through automatic analysis of system logs. This analysis is based on the assumption that the onus of miscommunication is on the system. Thus, instances of non-understandings, implicit and explicit confirmations based on false assumptions, and confusing prompts are treated as problematic system actions that we want to detect in order to avoid them. Since our main goal is to integrate the approach in a toolkit for offline analysis of interaction logs we focus here largely on models for offline detection. For this analysis, we have the full dialogue context (backward and forward) at our disposal, and use features that are both automatically extractable from the system logs and manually annotated. However, we also report the performances of these models using only online features and limited dialogue context, and demonstrate our models’ suitability for online use in detection of potential problems in system actions.

We evaluate our approach on datasets from three different dialogue systems that vary in their dialogue modeling, dialogue strategy, language, user types. We also report findings from an experimental work on cross-corpus analysis: using a model trained on logs from one system for analysis of interaction logs from another system. Thus the novelty of work reported here lies in our models’ relevance for offline as well as online detection of miscommunications, and the applicability and generalizability of features across dialogue systems and domains.

The paper is structured as follows: we report the relevant literature in Section 2 and establish the ground for our work. In Section 3 we describe the three datasets used. The annotation scheme is discussed in Section 4. The complete set of features explored in this work is presented in Section 5. The experimental method is described in Section 6 and results are reported in Section 7. We conclude and outline our future work in Section 8.

## 2 Background

One way to analyze miscommunication is to make a distinction between *non-understanding* and *misunderstanding* (Hirst et al., 1994). While non-understandings are noticed immediately by the listeners, the information about misunderstandings may surface only at a later stage in the dialogue. This can be illustrated with the following human-machine interaction:

1	S: <i>How may I help you?</i>
2	U: Can you recommend a Turkish restaurant in downtown area?
3	S: <i>Could you please rephrase that?</i>
4	U: A Turkish restaurant in downtown.
5	S: <i>Clowns, which serves Italian food, is a great restaurant in downtown area.</i>
6	U: I am looking for a Turkish restaurant

Table 1: An illustration of miscommunication in human-machine interaction. S and U denote system and user turns respectively. User turns are transcriptions.

The system, in turn 3, expresses that a non-understanding of user intentions (in turn 2) has occurred. In contrast, in turn 5 – following the best assessment of user turn 4 – the system makes a restaurant recommendation, but misunderstands the user’s choice of cuisine. However, this problem does not become evident until turn 6. The various approaches to detection of errors presented in the literature can be broadly classi-

fied in two categories – *early error detection* and *late error detection* – based on at what turns in the dialogue the assessments about errors are made (Skantze, 2007). In early error detection approaches the system makes an assessment of its current hypothesis of what the user just said. Approaches for detection of non-understanding, such as confidence annotation (Bohus & Rudnicky, 2002), fall in this category. In contrast, late error detection aims at finding out whether the system has made false assumptions about user’s intentions in previous turns. These distinctions are vital from our viewpoint as they point out the turns in dialogue that are to be assessed and the scope of dialogue context that could be exploited to make such an assessment.

We now present some of the related works and highlight what has been modeled as error, stage in dialogue the assessment about errors are made, and type of features and span of dialogue context used. Following this we discuss the motivations and distinct contributions of our work.

Walker et al. (2000) presented a corpus based approach that used information from initial system-user turn exchanges alone to forecast whether the ongoing dialogue will fail. If the dialogue is likely to fail the call could be transferred to a human operator right away. A rule learner, RIPPER (Cohen, 1995), was trained to make a forecast about dialogue failure after every user turn. The model was trained on automatically extracted features from automatic speech recognizer (ASR), natural language understanding (NLU) and dialogue management (DM) modules.

Bohus & Rudnicky (2002) presented an approach to utterance level *confidence annotation* which aims at making an estimate of the system’s understanding of the user’s utterance. The model returns a confidence score which is then used by the system to select appropriate dialogue strategy, e.g. express non-understanding of user intention. The approach combines features from ASR, NLU and DM for determining the confidence score using logistic regression.

Schmitt et al. (2011) proposed a scheme to model and predict the quality of interaction at arbitrary points during an interaction. The task for the trained model was to predict a score, from 5 to 1 indicating very high to very poor quality of interaction, on having seen a system-user turn exchange. A Support Vector Machine model was trained on automatically extractable features from ASR, NLU and DM modules. They observed that additional information such as user’s

affect state (manually annotated) did not help the learning task.

In their investigations of a Dutch Train timetable corpus, Krahmer et al., (2001) observed that dialogue system users provide positive and negative cues about misunderstandings on the system’s part. These cues include user feedback, such as corrections, confirmations, and marked disconfirmations, and can be exploited for late error detection.

Swerts et al. (2000) trained models for automatic prediction of user corrections. They observed that user repetition (or re-phrasing) is a cue to a prior error made by the system. They used prosodic features and details from the ASR and the DM modules to train a RIPPER learner. Their work highlights that user repetitions are useful cue for late error detection.

For our task, we have defined the problem as detecting miscommunication on the system’s part. This could be misunderstandings, implicit and explicit confirmations based on false assumptions, or confusing system prompts. Since instances of non-understandings are self-evident cases of miscommunication we exclude them from the learning task. Detecting the other cases of miscommunications is non-trivial as it requires assessment of user feedback. The proposed scheme can be illustrated in the following example interaction:

1	S: <i>How may I help you?</i>
2	U: Sixty One D
3	S: <i>The 61C. What’s the departure station?</i>
4	U: No

Table 2: An implicit confirmation based on false assumption is an instance of problematic system action. User turns are manual transcriptions

In the context of these four turns our task is to detect whether system turn 3 is problematic. If we want to use the model online for early error detection, the system should be able to detect the problem using only automatically extractable features from turn 1-3. Unlike *confidence annotation* (Bohus & Rudnicky, 2002), we also include what the system is about to say in turn 3 and make an anticipation (or forecast) of whether this turn would lead to a problem. Thus, it is possible for a system that has access to such a model to assess different alternative responses before choosing one of them. Besides using details from ASR and SLU components (exploited in the reported literature) the proposed *early model* is

able to use details from Dialogue Manager and Natural Language Generation modules.

Next, we train another model that extends the anticipation model by also considering the user feedback in turn 4, similar to Krahmer et al., (2001) and Swerts et al. (2000). Such a model can also be used online in a dialogue system in order to detect errors after-the-fact, and engage in late error recovery (Skantze, 2007). The end result is a model that combines both anticipation and user feedback to make an assessment of whether system turns were problematic. We refer to this model as the *late model*.

Since both the early and late models are to be used online, they only have access to automatically extractable features. However, we also train an *offline model* that can be used by a dialogue designer to find potential flaws in the system. This model extends the late model in that it also has access to features that are derived from manual annotations in the logs.

In this work we also investigated whether models trained on logs of one system can be used for error detection in interaction logs from a different dialogue system. Towards this we trained our models on generic features and evaluated our approach on system logs from three dialogue systems that differ in their dialogue strategy.

### 3 Corpora

Dialogue system logs from two publicly available corpora and one from a commercially deployed system were used for building and evaluating the three models. The first dataset is from the CamInfo Evaluation Dialogues corpus. The corpus comprises of spoken interactions between the Cambridge Spoken Dialogue System and users, where the system provides restaurant recommendations for Cambridge. The dialogue system is a research system that uses dialogue-state tracking for dialogue management (Jurcicek et al., 2012). As the system is a research prototype, users of these systems are not real users in real need of information but workers recruited via the Amazon Mechanical Turk (AMT). Nevertheless, the dialogue system is state-of-the-art in statistical models for dialogue management. From this corpus 179 dialogues were used as the dataset, which we will refer to as the **CamInfo** set.

The second corpus comes from the **Let’s Go** dialogue system. Let’s Go (Raux et al., 2005) is developed and maintained by the Dialogue Research Center (DialRC) at Carnegie Mellon University that provides bus schedule information

for Pittsburgh’s Port Authority buses during off-peak hours. The users of Let’s Go system are real users, which are in real need of the information. This makes the dataset interesting for us. The dataset used here consists of 41 dialogues selected from the data released for the 2010 Spoken Dialogue Challenge (Black et al., 2010).

The third dataset, **SweCC** – Swedish Call Center Corpus, is taken from a corpus of call logs from a commercial customer service provider in Sweden providing services in various domains. The system tries to extract some details from customers before routing the call to a human operator in the concerned department. Compared to CamInfo and Let’s Go datasets, the SweCC corpus is from a commercially deployed system, with real users, and the interactions are in Swedish. From this corpus 219 dialogues were selected. Table 3 provides a comparative summary of the three datasets.

CamInfo	Let’s Go	SweCC
Research	Research	Commercial
Hired users	Real users	Real users
Mostly implicit confirmation	Mostly explicit confirmation	Only explicit confirmation
Stochastic	Rule based	Rule based
English	English	Swedish
179 dialogues	41 dialogues	219 dialogues
5.2 exchanges on average per dialogue	19 exchanges on average per dialogue	6.6 exchanges on average per dialogue

Table 3: A comparative summary of the three datasets

## 4 Annotations

We take a supervised approach for detection of problematic system turns in the system logs. This requires each system turn in the training datasets to be labeled as to whether they are PROBLEMATIC (if the system turn reveals a miscommunication) or NOT-PROBLEMATIC. There are different schemes for labeling data. One approach is to ask one or two experts (having knowledge of the task) to label data and use inter-annotator agreement to set an acceptable goal for the trained model. Another approach is to use a few non-experts but use a set of guidelines so that the annotators are consistent (and to achieve a higher Kappa score, (Schmitt et al., 2011)). We take the crowdsourcing approach for annotating the CamInfo data and use the AMT platform. Thus, we avoid using both experts and guidelines. The key however is to make the task simple for the AMT-workers. Based on our earlier discussion on the role of dialogue context and type of errors

assessed in early and late error detection, we set up the annotation tasks such that AMT workers saw two dialogue exchanges (4 turns in total), as shown in Table 2:. The workers were asked to label system turn 3 as PROBLEMATIC or NOT-PROBLEMATIC, depending on whether it was appropriate or not, or PARTIALLY-PROBLEMATIC when it is not straightforward to choose between the former two labels.

In the Let’s Go dataset we observed that whenever the system engaged in consecutive confirmation requests the automatically extracted sub-dialogue (any four consecutive turns) did not always result in a meaningful sub-dialogue. Therefore the Let’s Go data was annotated by one of the co-authors of the paper. The SweCC data could not be used on AMT platform due to the agreement with the data provider, and was annotated by the same co-author. See Appendix A for sample of annotated interactions.

Since we had access to the user feedback to the questionnaire for the CamInfo Evaluation Dialogues corpus, we investigated whether the problematic turns identified by the AMT-workers reflect the overall interaction quality, as experienced by the users. We observed a visibly strong correlation between the user feedback and the fractions of system turn per dialogue labeled as PROBLEMATIC by the AMT-workers. Figure 1 illustrates the correlation for one of the four questions in the questionnaire. This shows that the detection and avoidance of problematic turns (as defined here), will have bearing on the users’ experience of the interaction.

Each system turn in the CamInfo dataset was initially labeled by two AMT-workers. In case of a tie, one more worker was asked to label that instance. In total 753 instances were labeled in the first step. We observed an inter-annotators agreement of 0.80 (Fleiss Kappa) among the annotators and only 113 instances had a tie and were annotated by a third worker. The label with the majority vote was chosen as the final class label for instances with ties in the dataset. Table 4 shows the distributions for the three annotation categories seen in the three datasets. Due to the imbalance of the PARTIALLY-PROBLEMATIC class in the three datasets we excluded this class from the learning task and focus only on classifying system turns as either PROBLEMATIC or NOT-PROBLEMATIC. System turns expressing non-understanding were also excluded from the learning task. The final datasets had the following representation for the PROBLEMATIC class: CamInfo (615) 86.0%, Let’s Go (744) 57.5, and

for SweCC (871) 65.7%. To mitigate the high class imbalance in CamInfo another 51 problematic dialogues (selected following the correlations of user feedback from Figure 1) were annotated by a second co-author. The resulting CamInfo dataset had 859 instances of which 75.3% are from PROBLEMATIC class.

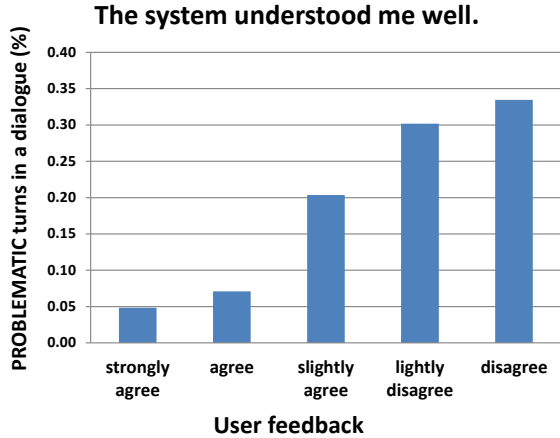


Figure 1: Correlation of system turns annotated as problematic with user feedback

Dataset (#instances)	CamInfo (753)	Let’s Go (760)	SweCC (968)
PROBLEMATIC	16 %	42%	31%
NOT-PROBLEMATIC	73 %	57%	61%
PARTIALLY-PROBLEMATIC	11 %	1%	8%

Table 4: Distribution of the three annotation categories across the three datasets

## 5 Features

We wanted to train models that are generic and can be used to analyze system logs from different dialogue systems. Therefore we trained our models on only those features that were available in all the three datasets. Below we describe the complete feature set, which include features and manual annotations that were readily available in system logs. A range of higher-level features were also derived from the available features. Since the task of the three dialogue system is to perform slot-filling we use the term *concept* to refer to slot-types and slot-values.

**ASR:** the best hypothesis, the recognition confidence score and the number of words. **NLU:** user dialogue act (the best parse hypothesis – **nlu\_asr**), the best parse hypothesis obtained on manual transcription (**nlu\_trn**), number of concepts in **nlu\_asr** and **nlu\_trn**, concept error rate: the Levenshtein distance between **nlu\_asr** and **nlu\_trn**, correctly transferred concepts: the

fraction of concepts in **nlu\_trn** observed in **nlu\_asr**. **NLG:** system dialogue act, number of concepts in system act, system prompt, and number of words in the prompt.

**Manual annotations:** manual transcriptions of the best ASR hypothesis, number of words in the transcription, word error rate: the Levenshtein distance between the recognized hypothesis and transcribed string, correctly transferred words: fraction of words in the transcription observed in the ASR hypothesis.

**Discourse features:** *position in dialogue:* fraction of turns completed up to the decision point. **New information:** fraction of new words (and concepts) in the successive prompts of a speaker. **Repetition:** Two measures to estimate repetition in successive speaker turns were used: (i) cosine similarity, the cosine angle between vector representation of the two turns and (ii) the number of common concepts. **Marked disconfirmation:** whether the user response to a system request for confirmation has a marked disconfirmation (e.g., “no”, “not”). **Corrections:** the number of slots-values in previous speaker turn that were given a new value in the following turn – by either the dialogue partner or the same speaker – were used as an estimate of user corrections, false assumptions and rectifications by the system, and change in user intentions.

## 6 Models and Method

As mentioned earlier, the *early* and *late* models are aimed at online use in dialogue systems, whereas the *offline model* is for offline analysis of interaction logs. A window of 4 turns, as discussed in Section 2, is used to limit the dialogue context for extraction of features. Accordingly, the *early model* uses features from turns 1-3; the *late model* uses features from the complete window, turns 1-4. The *offline model* like the *late model* uses the complete window, but additionally uses the manual transcription features or features derived from them, e.g. word error rate.

For the purpose of brevity, we report four sets of feature combinations: (i) Bag of words representation of system and user turns (BoW), (ii) DrW: a set containing all the features derived from the words in the user and system turns, e.g., turn length (measured in number of words), cosine similarity in speaker turns as an estimate of speaker repetition, (iii) Bag of concept representation of system and user dialogue acts (BoC), and (iv) DrC: a set with all the features derived

from dialogue acts, e.g., turn length (measured in number of concepts).

Given the skew in distribution of the two classes in the three datasets (cf. Section 4) accuracy alone is not a good evaluation metric. A model can achieve high classification accuracy by simply predicting the value of the majority class (i.e. NOT-PROBLEMATIC) for all predictions. However, since we are equally interested in the recall for both PROBLEMATIC and NOT-PROBLEMATIC classes, we use the un-weighted average recall (UAR) to assess the model performance, similar to Higashinaka et al., 2010).

We explored various machine learning algorithms available in the Weka toolkit (Hall et al., 2009), but report here models trained using two different algorithms: JRIP, a Weka implementation of the RIPPER rule learning algorithm, and Support Vector Machine (SVM) with linear kernel. The rules learned by JRIP offer a simple insight into what features contribute in decision making. The SVM algorithm is capable of transforming the feature space into higher dimensions and learns sophisticated decision boundaries. The figures reported here are from a 10-fold cross-validation scheme for evaluation.

## 7 Results

### 7.1 Baseline

To assess the improvements made by the trained models we need a baseline model to draw comparisons. We can use the simple majority class baseline model that will predict the value of majority class for all predictions. The UAR for such a model is shown in Table 5 (row 1). The UAR for all the three datasets is 0.50.

All the three dialogue systems employ confirmation strategies, which are simple built-in mechanisms for detecting miscommunication online. Therefore, a model trained using the *marked disconfirmation* feature alone could be a more reasonable baseline model for comparison. Row 2 in Table 5 (feature category *MDisCnf*) shows the performances for such a baseline. The figures from *late* and *offline* models suggest that while this feature is not at all useful for CamInfo dataset (UAR = 0.50 for both JRIP and SVM) it makes substantial contributions to models for Let's Go and SweCC datasets. The *late model*, using the online features for marked disconfirmation and the JRIP algorithm obtained a UAR of 0.68 for Let's Go and 0.87 for SweCC. The corresponding *offline* models, which use the manual feature in addition, achieve even better results for

the two datasets: UAR of 0.74 and 0.89 respectively. These figures clearly illustrate two things: First, while Let's Go and SweCC systems often employ explicit confirmation strategy, CamInfo hardly uses it. Second, the majority of problems in the Let's Go and SweCC are due to explicit confirmations based on false assumptions.

### 7.2 Word-related features

Using the bag of word (BoW) feature set alone, we observe that for CamInfo dataset the SVM achieved a UAR of 0.75 for the *early* model, 0.79 for the *late* model, and 0.80 for the *offline* model. These are comprehensive gains over the baseline of 0.50. The figures for the *early* model suggest that by looking only at (i) the most recent user prompt, (ii) the system prompt preceding it, and (ii) the current system prompt which is to be executed, the model can anticipate, well over chance whether the chosen system prompt would lead to a problem.

For the Let's Go and SweCC datasets, using the BoW feature set the *late model* achieved modest gains in performance over the corresponding MDisCnf baseline model. For example, using the SVM algorithm the *late model* for Let's Go achieved a UAR of 0.81. This is an absolute gain of 0.13 points over the UAR of 0.68 achieved using the marked disconfirmation feature set alone. This large gain can be attributed partly to the *early* model (a UAR of 0.74) and the late error detection features which add another 0.07 absolute points raising the UAR to 0.81. For the SweCC dataset, although the gains made by the JRIP learner models over the MDisCnf baseline are marginal, the fact that the *late model* gains in UAR scores over *early* model points to the contributions of words that indicate user disconfirmations, e.g. *no* or *not*.

Next, on using BoW feature set in combination with the DrW feature set that contains features derived from words, such as prompt length (number of words), speaker repetitions, ASR confidence score, etc., we achieved both minor gains and losses for the CamInfo and Let's Go dataset. The *offline* models for Let's Go (both JRIP as well as SVM) made a gain of approx. 0.04 over the *late* models. A closer look at the rules learned by the JRIP model indicates that features such as word error rate, cosine similarity measure of user repetition, number of words in user turns, contributed to rule learning.

In the SweCC dataset we observe that for all the *early* and *late* models the combination of BoW and DrW feature sets offered improved

SNr.			CamInfo		Let's Go		SweCC	
			UAR		UAR		UAR	
1.	Majority class baseline		0.50			0.50	0.50	
	Feature Set	Model	JRip	SVM	JRip	SVM	JRip	SVM
2.	MDisCnf	Late	0.50	0.50	0.68	0.68	0.87	0.83
		Offline	0.50	0.50	0.74	0.73	0.89	0.84
3.	BoW	Early	0.72	0.75	0.72	0.74	0.78	0.80
		Late	0.73	0.79	0.80	0.81	0.88	0.88
		Offline	0.78	0.80	0.84	0.82	0.90	0.89
4.	BoW+DrW	Early	0.75	0.77	0.71	0.75	0.84	0.82
		Late	0.71	0.82	0.82	0.80	0.92	0.91
		Offline	0.77	0.79	0.85	0.84	0.92	0.90
5.	BoC	Early	0.80	0.81	0.76	0.76	0.81	0.81
		Late	0.81	0.82	0.86	0.84	0.89	0.88
		Offline	0.81	0.82	<b>0.88</b>	0.85	-	-
6.	BoC+DrC+DrW	Early	0.80	0.83	0.70	0.80	0.84	0.82
		Late	0.78	0.82	0.84	0.85	<b>0.93</b>	0.89
		Offline	0.82	<b>0.84</b>	0.87	0.86	0.92	0.89

Table 5 : Performance of the various *early*, *late* and *offline* models for error detection on the three datasets

performances over using BoW alone. The rules learned by the JRIP indicate that in addition to the marked disconfirmation features the model is able to make use of features that indicate whether the system takes the dialogue forward, the ASR confidence score for user turns, the position in dialogue, and the user turn lengths.

### 7.3 Concept-related features

Next, we analyzed the model performances using the bag of concept (BoC) feature set alone. A cursory look at the performances in row 5 in Table 5 suggest that for both CamInfo and Let's Go the BoC feature set offers modest and robust improvement over using BoW feature set alone. In comparison, for the SweCC dataset the gains made by the models over using BoW alone are marginal. This is not surprising given the high UARs achieved for SweCC corresponding to the MDisCnf feature set (row 2), suggesting that most problems in SweCC dataset are inappropriate confirmation requests, and detection of user disconfirmations is a good enough measure.

We also observed that the contribution of the *late* model is much clearly seen in Let's Go and SweCC datasets while this is not true for CamInfo. In view of the earlier observation that explicit confirmations are seldom seen in CamInfo we can say that users are left to use strategies such as repetitions to correct false assumptions by the system. These cues of corrections are much harder to assess than the marked disconfirmations. The best performances were in general obtained by the *offline* models: UAR of 0.82 on CamInfo dataset using SVM algorithm and 0.88

for Let's Go using JRIP. Some of the features used by the JRIP rule learner include: number of concepts in parse hypothesis being zero, the system dialogue act indicating open prompts "How may I help you?" during the dialogue (suggesting a dialogue restart), and slot types which the system often had difficulty understanding. These were user requests for price range and postal codes in the CamInfo dataset, and time of travel and place of arrival in the Let's Go dataset. As the NLU for manual transcription is not available for the SweCC dataset the corresponding row for the *offline* model in Table 5 is empty.

Next, we trained the models on the combined feature set, i.e. BoC, DrC and DrW sets. We observed that while majority of models achieved marginal gains over using BoC set alone, the ones that did lose did not exhibit a major drop in performance. The best performance for the CamInfo is obtained by the *offline* model (using the SVM algorithm): a UAR of 0.84. For Let's Go the JRIP model achieved the best UAR, 0.87 for the *offline* model. For the SweCC the *late* model performed better than the *offline* model and achieved a UAR of 0.93 using the JRIP learner. These are comprehensive gains over the two baseline models. Appendix A shows two examples of offline error detection.

### 7.4 Impact of data on model performances

We also analyzed the impact of amount of training data used on model performances. A hold-out validation scheme was followed. A dataset was first randomized and then split into 5 sets, each containing equal number of dialogues. Each of

the set was used as a hold-out test set for models trained on the remaining 4 sets. Starting with only one of the 4 sets as the training set, four rounds of training and testing were conducted. At each stage one whole set of dialogue was added to the existing training set. The whole exercise was conducted 5 times, resulting in a total of  $5 \times 5 = 25$  observations per evaluation. Each point in Figure 2 illustrates the UAR averaged over these 25 observations by the *offline* model (JRIP learner using feature set 6, cf. row 6 in Table 5). The performance curves and their gradients suggest that all the models for the three datasets are likely to benefit from more training data, particularly the CamInfo dataset.

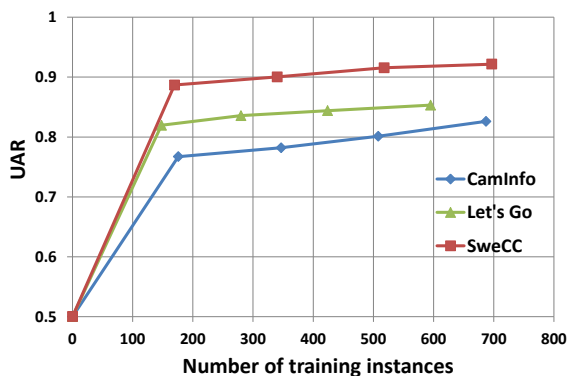


Figure 2: Gains in UAR made by the *offline* model (JRIP learner and feature set BoC+ DrW+DrC)

Training set →	CamInfo	Let's Go	SweCC
Test set	UAR	UAR	UAR
CamInfo	-	0.72	0.54
Let's Go	0.62	-	0.73
SweCC	0.53	0.89	-

Table 6: Cross-corpus performances of *offline* model (JRIP learner and feature set BoC+ DrW+DrC)

### 7.5 A model for cross-corpus analysis

We also investigated whether a model trained on annotated data from one dialogue system can be used for automatic detection of problematic system turns in interaction logs from another dialogue system. Table 6 illustrates the performances of the *offline* model (JRIP learner using feature set 6, cf. row 6 in Table 5). This experiment mostly used numeric features such as turn length, word error rate, and dialogue act features that are generic across domains, e.g., request for information, confirmations, and disconfirmations.

We observed that using the Let's Go dataset as the training set we can achieve a UAR of 0.89 for SweCC and 0.72 for CamInfo. Although both SweCC and Let's Go use explicit clarifications, since SweCC dataset exhibits limited error pat-

terns a UAR of only 0.73 is obtained for Let's Go when using a model trained on SweCC. Models trained on CamInfo seem more appropriate for Let's Go than for SweCC.

## 8 Conclusions and Future work

We have presented a data-driven approach to detection of problematic system turns by automatic analysis of dialogue system interaction logs. Features that are generic across dialogue systems were automatically extracted from the system logs (of ASR, NLU and NLG modules) and the manual transcriptions. We also created abstract features to estimate discourse phenomena such as user repetitions and corrections, and discourse progression. The proposed scheme has been evaluated on interaction logs of three dialogue systems that differ in their domain of application, dialogue modeling, dialogue strategy and language. The trained models achieved substantially better recall on the three datasets. We have also shown that it is possible to achieve reasonable performance using models trained on one system to detect errors in another system.

We think that the models described here can be used in many different ways. A simple application of the online models could be to build an "error awareness" module in a dialogue system. For offline analysis, the late error detection model could be trained on a subset of data collected from a system, and then applied to the whole corpus in order to find problematic turns. Then only these turns would need to be transcribed and analyzed further, reducing a lot of manual work. However, we also plan in a next step to not only find instances of miscommunication automatically, but also summarize the main root causes of the problems, in order to help the dialogue designer to mitigate them. This could include extensions of grammars and vocabularies, prompts that need rephrasing, or lack of proper error handling strategies.




### Acknowledgement

We would like to thank our colleagues Giam-piero Salvi and Kalin Stefanov for their valuable discussions on machine learning. We also want to thank the CMU and Cambridge research groups for making the respective corpus publicly available. This research is supported by the EU project SpeDial – Spoken Dialogue Analytics, EU grant # 611396.



## Reference

- Black, A. W., Burger, S., Langner, B., Parent, G., & Eskenazi, M. (2010). Spoken Dialog Challenge 2010.. In Hakkani-Tür, D., & Ostendorf, M. (Eds.), *SLT* (pp. 448-453). IEEE.
- Bohus, D., & Rudnicky, A. (2002). *Integrating multiple knowledge sources for utterance-level confidence annotation in the CMU Communicator spoken dialog system*. Technical Report CS-190, Carnegie Mellon University, Pittsburgh, PA.
- Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).
- Higashinaka, R., Minami, Y., Dohsaka, K., & Meguro, T. (2010). Modeling User Satisfaction Transitions in Dialogues from Overall Ratings. In *Proceedings of the SIGDIAL 2010 Conference* (pp. 18-27). Tokyo, Japan: Association for Computational Linguistics.
- Hirst, G., McRoy, S., Heeman, P., Edmonds, P., & Horton, D. (1994). Repairing conversational misunderstandings and non-understandings. *Speech Communication*, 15, 213-230.
- Jurcicek, F., Thomson, B., & Young, S. (2012). Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech & Language*, 26(3), 168-192.
- Krahmer, E., Swerts, M., Theune, M., & Weegels, M. (2001). Error detection in spoken human-machine interaction. *International Journal of Speech Technology*, 4(1), 19-29.
- Raux, A., Langner, B., Bohus, D., Black, A. W., & Eskenazi, M. (2005). Let's go public! Taking a spoken dialog system to the real world.. In *INTER-SPEECH* (pp. 885-888). ISCA.
- Schmitt, A., Schatz, B., & Minker, W. (2011). Modeling and Predicting Quality in Spoken Human-computer Interaction. In *Proceedings of the SIGDIAL 2011 Conference* (pp. 173-184). Stroudsburg, PA, USA: Association for Computational Linguistics.
- Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems*. Doctoral dissertation, KTH.
- Swerts, M., Hirschberg, J., & Litman, D. (2000). Corrections in spoken dialogue systems. In *Proceedings of the International Conference on Spoken Language Processing*. Beijing, China.
- Walker, M. A., Langkilde, I., Wright, J., Gorin, A., & Litman, D. J. (2000). Learning to predict problematic situations in a spoken dialogue system: experiments with How may I help you?. In *Proceedings of North American Meeting of the Association of Computational Linguistics*.

**Appendix A:** Table 7 and Table 8 illustrate the predictions made by the *offline* model using JRIP rule learner and the combined features: BoC+DrW+DrC (cf. row 6, Table 5). Colum headings T and P contain the training and predicted labels, respectively.  is PROBLEMATIC,  is NOT-PROBLEMATIC, and  is an instance of non-understanding therefore excluded from the learning task. Table 9 illustrates the top rules learned by the JRIP learner, for each of the two datasets.









	T	P
1 S: <i>Welcome to Cambridge system. How may I help you?</i>		
2 U: I want to find a [I want to find a pub]		
3 S: <i>The Avery is a great pub.</i>		
4 U: What is the phone number [what is the phone number]		
5 S: <i>The phone number of The Avery is 01223 323405.</i>		
6 U: Does it have internet [Uh uh have in there]		
7 S: <i>Can I help you with anything else?</i>		
8 U: Does that pub have internet [Where is that er have in]		
9 S: <i>The Avery is on 69-73 Regent Street.</i>		
10 U: Thank you bye.		

Table 7: Offline error detection on a CamInfo interaction. Text within [] is the top ASR hypothesis.













7 S: <i>What can I do for you?</i>		
8 U: Next bus [65e]		
9 S: <i>The 65E. Did I get that right?</i>		
10 U: No [no]		
11 S: <i>What can I do for you?</i>		
12 U: 61A [61A]		
13 S: <i>The 61A. Did I get that right?</i>		
14 U: Yes [yes]		
15 S: <i>Alright Where would you like to leave from?</i>		
16 U: Atwood [Duquensne]		
17 S: <i>For example, you can say, NORTH BRADDOCK, FORBES AND MURRAY, or DOWNTOWN.</i>		
18 U: Forbes and Atwood [Forbes big at Atwood in wall]		
19 S: <i>ATWOOD IN WALL. Did I get that right?</i>		
20 U: No [no]		

Table 8: Offline error detection on a Let’s Go interaction. Text within [] is the top ASR hypothesis

<i>The top rule learned by JRIP on CamInfo dataset</i>	<i>The top 2 rules learned by JRIP on Let’s Go dataset</i>
<p>1. (ctc-svp-usr-2 &lt;= 0.5) and (frc-new-slt-asr-usr-2 &lt;= 0.5) =&gt;class=problematic (83.0/9.0)</p> <p><b>Rule 1:</b> If the correctly transferred concept rate for user turn 2 is &lt;= 0.5 and the number of new slots mentioned are &lt;= 0.5 then <i>anticipate</i> the system turn 3 as PROBLEMATIC. A total of 83 instances were labeled problematic by this rule, 9 of which were false predictions.</p> <p><b>Summary:</b> The user repeats (rephrases) to correct the system’s mistake in grounding. However, the system does not have a good model to detect this and therefore the system response is most likely to be perceived in appropriate by the user.</p>	<p>1. (wer-tr-usr-2 &gt;= 20) and (4-dact-tr_no &gt;= 1) =&gt; class=problematic (121.0/3.0)</p> <p>2. (ctc-svp-usr-2 &lt;= 0.5) and (4-dact-tr_yes &lt;= 0) =&gt; class=problematic (115.0/23.0)</p> <p><b>Rule 1:</b> If WER for user turn 2 is more than 20 and the user d-act in turn 4 is “no” then the system response in turn 3 was PROBLEMATIC.</p> <p><b>Rule 2:</b> Similar to the Rule 1 but uses different features. If correctly transferred concept rate for user turn 2 is &lt;= 0.5 and in turn 4 the user act was not “yes” then the system action in turn 3 was PROBLEMATIC.</p> <p><b>Summary:</b> Model uses late error detection cues such as marked disconfirmations to assess system actions.</p>

Table 9: The top rules learned by the JRIP model for *offline* error detection on the CamInfo and Let’s Go datasets (cf. row 6, Table 5).