# Integrating Multiple Knowledge Sources for Utterance-Level Confidence Annotation in the CMU Communicator Spoken Dialog System

Dan Bohus Alex Rudnicky

November 2002
CMU-CS-02-190

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Abstract**

In the recent years, automated speech recognition has been the main drive behind the advent of spoken language interfaces, but at the same time a severe limiting factor in the development of these systems. We believe that increased robustness in the face of recognition errors can be achieved by making the systems aware of their own misunderstandings, and employing appropriate recovery techniques when breakdowns in interaction occur. In this paper we address the first problem: the development of an utterance-level confidence annotator for a spoken dialog system. After a brief introduction to the CMU Communicator spoken dialog system (which provided the target platform for the developed annotator), we cast the confidence annotation problem as a machine learning classification task, and focus on selecting relevant features and on empirically identifying the best classification techniques for this task. The results indicate that significant reductions in classification error rate can be obtained using several different classifiers. Furthermore, we propose a data driven approach to assessing the impact of the errors committed by the confidence annotator on dialog performance, with a view to optimally fine-tuning the annotator. Several models were constructed, and the resulting error costs were in accordance with our intuition. We found, surprisingly, that, at least for a mixed-initiative spoken dialog system as the CMU Communicator, these errors trade-off equally over a wide operating characteristic range.

This research was sponsored in part by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

**Keywords:** spoken dialog systems, utterance-level confidence annotation, features and classification techniques for confidence annotation, modelling costs of misunderstandings.

# 1 Introduction

Over the recent years, speech recognition technology has been making steady and significant progress. Together with advances in robust parsing techniques, natural language generation algorithms, and the advent of high-quality speech synthesis systems, it has paved the way for the emergence of complex, interactive spoken dialog systems.

A large number of these systems are currently under development in various universities and research labs across the world: Communicator [1, 2, 3] (CMU and others), Jupiter [4], Mercury [5] (MIT), ELVIS [8], How May I Help You? [9] (AT&T) , TRAINS/TRIPS [11] (Rochester University), WAXHOLM [10] (TMH), to name just a few. Other systems have already successfully transitioned into day-to-day use – for instance, United Airlines makes use of a spoken dialog system called SIMON to handle claims for missing and delayed luggage. Companies like TellMe [33], BeVocal [34], HeyAnita [35] successfully provide various platforms and applications to deliver voice dialing, messaging and a large diversity of customer care services. These systems operate across a variety of domains, but, given the limitations in subjacent technologies, most of them fall into one of two categories: they either act like spoken language front-ends for an information access system, or as simple spoken language command-and-control interfaces to various devices. More recently, new efforts aim at the development of other, more sophisticated types of natural language based interactive agents, like personal assistants, taskable agents, interactive tutors, call-routing systems, embodied conversational agents, etc.

Automatic recognition of spontaneous speech is still imperfect at best, and, although it is the main drive behind the advent of spoken language interfaces, it also is one of the most severe limiting factors in the development of these systems. Small changes in the environment, microphone or telephone line quality have a great impact on, and can seriously impair recognition performance. Moreover, as spoken dialog systems are in general targeted to large populations, speaker variability (i.e. pronunciation style, native and non-native accents, etc) represents another major concern. When adding to this mix the disfluencies (stutters, restarts, false-starts, coughs and various other acoustical non-lexical events) characteristic to spoken language, the recognition error rates jump from a typical 8-10% for read speech to 20-30% for spontaneous speech.

The impact of recognition errors on a spoken dialog system is in most cases significant. Typically, the recognition errors will propagate to the upper levels of the system - language understanding (semantic parsing) and dialog management. If the parser lacks robustness, even the smallest recognition error can ruin the whole language understanding process: if no semantic representation of the input can be generated, we have a *non-understanding* error. Moreover, even if the parsing succeeds, the misrecognition may still persist in the parse and will be propagated to the dialog management level - in this case we are dealing with a *misunderstanding*. The system will generally take the misunderstanding as fact and will act using invalid information. When this happens, a simple recognition error can completely compromise the success of the interaction, as the example in Figure 1 illustrates. The system is unaware of the problem because it blindly trusts its inputs, and has no means to judge how well the conversation is proceeding.

One approach to addressing the problem of recognition errors is therefore to increase the robustness of these systems when faced with misunderstandings. We believe that this can be achieved by creating an effective performance monitoring and recovery process: after all, human-human conversations are also affected by misunderstandings, but we typically manage to reestablish a mutual ground by employing various confirmation and clarification

This sample conversation was collected from the CMU Communicator [1, 2], a spoken dialog system operating in the air-travel planning domain. The system's utterances are prefixed by `S:`, and the user's utterances by `U:`. The actual recognized utterances are also presented in square brackets. In turn 4, the user volunteers his name (omitted here for privacy reasons), but this is misrecognized by the system for [ATHENS IN AKRON]. This simple misunderstanding throws the system on a wrong path and ruins the whole interaction, to the point where the user decides to restart the whole conversation.

| | | |
|---|---|---|
| 1 | `S:` | `Are you a registered user ?` |
| 2 | `U:` | *No I'm not. No [NO I'M NOT NO]* |
| 3 | `S:` | `What is your full name ?` |
| 4 | `U:` | *User_Name [ATHENS IN AKRON]* |
| 5 | `S:` | `There is an Athens in Georgia and in Greece. Which destination did you want ?` |
| 6 | `U:` | *Neither [NEITHER]* |
| 7 | `S:` | `I'm sorry, I didn't catch that. There is an Athens in Georgia and in Greece. Which destination did you want ?` |
| 8 | `U:` | *Georgia [GEORGIA]* |
| 9 | `S:` | `A flight from Athens... Where do you want to go ?` |
| 10 | `U:` | *Start over [START OVER]* |
| 11 | `S:` | `Do you really want to start over ?` |
| 12 | `U:` | *Yes [YES]* |
| 13 | `S:` | `What city are you leaving from ?` |
| 14 | `U:` | *Hamilton, Ontario [HILTON ONTARIO]* |
| 15 | `S:` | `Sorry, I'm not sure I understood what you said. Where are you leaving from ?` |
| 16 | `U:` | *Hamilton [HILTON]* |
| 17 | `S:` | `I'm still having trouble understanding you. To go on, I need you to answer the following question. What city are you leaving from ?` |
| 18 | `U:` | *Toronto [TORONTO]* |

Figure 1: Sample session affected by misunderstandings in the CMU Communicator spoken dialog system

techniques. The two key ingredients needed to spawn this desired robustness are: (1) we need to make the systems aware of their own misunderstandings, and we need a quantitative assessment of the impact of these misunderstandings on dialog performance. Solving this first set of problems will serve as a basis for the second part of the solution, (2) the development of a set of appropriate recovery techniques that are to be used when such breakdowns in interaction are detected.

In this paper we address the first of these two problems: we present the development of an utterance-level confidence annotator for the Carnegie Mellon Communicator system, together with cost models for the errors it commits.

The confidence annotation problem has been investigated previously [12, 13, 14, 15]. Most of these efforts were focused on how to detect the decoding errors made by the speech recognizer outside the context of a spoken dialog system, and thus the proposed schemes work mainly at the frame, phoneme, or word level. For instance, word-level confidence annotation assigns a reliability tag to each word token in the decoder hypothesis. Typically a two-class annotation scheme is used, which marks the word instance correct or incorrect. However, this type of model is not always sufficient for dialog systems. A more desirable solution would be an annotator which integrates information from multiple knowledge sources existent in the dialog system, and which outputs a continuous score representing the degree of belief in correct perception of a certain user utterance.

More recently, attempts have been made to use features from the other levels of the dialog system in deriving confidence metrics. For example [16] reports a study using decoder, language model and parsing features with a neural network classifier. Others [17] have addressed the second part of our stated problem, and devised ways to use confidence metrics in the upper levels of language understanding and dialog management in order to achieve more flexible dialog and clarification strategies.

In this work, we also start by casting the problem of confidence annotation as a machine-learning task. A set of 12 relevant features from three different knowledge sources in the dialog system (speech recognition, parsing and dialog management) are identified and used to train several classifiers in an attempt to empirically establish which performs best in this task. Furthermore, in the second part of the paper (Section 4) we perform an empirical evaluation of the impact of confidence annotation errors on the system's performance. We then show how the costs of errors resulting from the constructed models could be used to optimally fine-tune the confidence annotator. In this process we obtain the notable result that, at least for a mixed-initiative spoken dialog system like the CMU Communicator, these errors trade-off almost equally over a wide operating range.

The rest of the paper is organized as follows: we continue with a brief introduction to the Carnegie Mellon Communicator spoken dialog system, as it sets the framework for our work. Then, in Section 3, we present the development of a binary utterance-level confidence annotator, and comment on its extension to a continuous, probability-score annotator. Section 4 addresses the issue of modelling the costs of the various types of errors that the developed annotator commits, with the view to fine-tuning its performance. Lastly, Section 5 summarizes the insights gained from this work, presents our conclusions and indicates several directions for future research.

## 2 The Carnegie Mellon Communicator

We briefly present the Carnegie Mellon Communicator spoken dialog system as it provides the framework and the target platform for the development of the utterance-level confidence annotator. The CMU Communicator is a telephone-based spoken dialog system that operates in the air-travel planning domain. It can engage in complex, mixed-initiative conversations with the users, gathering constraints, constructing multi-leg itineraries and handling local arrangements (i.e. hotel reservations and car rentals) for each of the visited cities. The system uses real-time flight information collected from the web and knows of about five hundred cities in the United States and abroad. For more specific information on the functionality of the CMU Communicator and its components, the interested reader is pointed to [1, 2, 3]. The source code for the CMU Communicator has recently been made publicly available under an open-source license (see `www.speech.cs.cmu.edu/Communicator`).

Structurally, the CMU Communicator is based on the Galaxy [19] architecture. The system is composed of a series of parallel modules (i.e. speech recognition, understanding, synthesis, dialog management, etc.) that interact with each other by means of a central programmable traffic-router - the Galaxy Hub. Below we briefly comment on the main components of this system, with a focus on aspects relevant to the construction of the utterance-level confidence annotator.

The CMU Communicator employs the SPHINX-II decoder [20] in real-time mode to perform continuous speech recognition. In the configuration we are currently using, SPHINX provides a limited form of word-level confidence annotation: each word token in the decoder hypothesis is marked as either being recognized with a high confidence rating or not. Upon completion of the decoding phase, the top-level hypothesis is passed to the PHOENIX parser [21], which, based on a hand-written semantic grammar, outputs the sequence of semantic concepts extracted from the utterance.

Subsequently, the parsing result is passed to HELIOS - the component responsible for detecting possible misunderstandings. Previous to this work, HELIOS employed a simple heuristic rule based on parsing coverage and fragmentation. When parse coherence was very low, this component raised a ***Garble*** flag, indicating to the upper levels of the dialog system that the current utterance is most likely a misunderstanding.

At the core of the CMU Communicator lies the AGENDA Dialog Manager [3], which controls the interaction with the user, steering the conversation appropriately. Its actions are controlled by a set of handlers that are dynamically ordered into an agenda, as the user advances through the task. Each (unblocked) handler is given the chance to attach to and consume concepts occurring in the input parse, and can trigger calls to other task agents (i.e. web-based flight retrieval), or can invoke the natural language generation component to provide a response to the user. In the current system, ***Garble*** utterances are handled by signaling the misunderstanding to the user, and repeating the previous system prompt.

The natural language generation component used in Communicator relies on a mixture of template-based and stochastic sentence generation [22]. Finally, the resulting prompt is synthesized using a limited domain unit selection synthesis, based on the Festival Speech Synthesis System [23].

In the next sections, we present the development and analysis of a new utterance-level confidence annotation mechanism, as a replacement and improvement over the rule-based misunderstanding detector previously employed by HELIOS.

# 3 Utterance Level Confidence Annotation

The problem of utterance-level confidence annotation can naturally be cast as a machine learning classification task: given the current user utterance, select a set of relevant features, and use them to classify the utterance as correctly understood or not. Note that there are at least two different flavors to this problem: on one hand, one could be interested only in accepting or rejecting an utterance, in which case the problem reduces to a binary classification task. Alternatively, one might be interested in obtaining a continuous score reflecting the probability of correct understanding – this could be used for instance for later integration in a Bayesian decision framework at the dialog management level. We start by addressing the first problem in Section 3.1; subsequently, in Section 3.2, we discuss in more detail how the presented approach can be extended to provide a continuous, probability-like score.

## 3.1 Binary Utterance Level Confidence Annotation

Once cast as a machine learning classification task, the problem can be decomposed into several parts: first, we need to collect and label a corpus of utterances, which will be used for training and testing the classifier. Next, we need to identify a set of features that are relevant for detecting misunderstandings. Last but not least, we need to make a decision with regard to which classifier is most appropriate for the given task.

In the following subsections we address in turn each of these issues, after which we present and comment on the results obtained in the various experiments performed.

### 3.1.1 Data collection

Two separate corpora were used throughout the experiments described in this paper. Initially, we started with Communicator logs and transcripts from an early version of the system (October and November of 1999), as that data was readily available. In the sequel, we will refer to this initial corpus as OctNov99.

A lot of changes have occurred since then: a new AGENDA dialog manager architecture was introduced in the system, as well as a new stochastic language generation component and state-specific language models; furthermore, new functionality for hotel and car reservations was added. We felt that the combined effect of these changes had a significant impact on the behavior of the system, and therefore we started another data collection effort, which resulted in a new, less noisy, and more up-to-date corpus, which we will hereafter refer to as JuneJuly01.

As the system was publicly available from its earliest stages of development, both corpora contain a mixture of developer and outsider calls. The JuneJuly01 corpus was however the result of a more directed data-collection effort. While in the first corpus most sessions were free dialogs, in the JuneJuly01 corpus the largest proportion of the sessions was scenario-based. Four different scenarios were employed to collect the data, varying across several dimensions, like number of legs for the trip, the locations involved, and the hotel and car requirements. Basic statistics for these corpora are illustrated in Table 1.

| **Corpus** | **OctNov99** | **JuneJuly01** |
|---|---|---|
| Dialog types | Developer and outsider calls | Mostly scenario-based calls |
| Number of dialogs | 311 | 134 |
| Number of utterances | 4550 | 2561 |

Table 1: Basic corpus statistics

Two components of these corpora are of special interest for the stated problem: the system logs and the utterance transcripts. The logs were automatically generated in each session by the various Communicator components involved, and served as the source of features for the classification process. The transcripts were created and double-checked by a human annotator, and were used to establish the appropriate confidence label, as described below.

Initially, both corpora were manually labelled for confidence at the concept-level. Labelling was performed by comparing the decoded hypothesis and parsing results with the reference transcript. Based on this comparison, each concept appearing in the parse was assigned one of 4 possible labels: `OK`, `RBAD`, `PBAD`, `OOD`. A concept would be labelled `OK`, if it was correctly perceived by the system, i.e. if both recognition and parsing were correct. The `RBAD` label was assigned to misunderstood concepts, when the misunderstanding was triggered by poor recognition. Similarly, the `PBAD` label was assigned to misunderstood concepts, when the error was triggered by an incorrect parse (i.e. insufficient grammar coverage). Finally, unparsed utterances which where not within the scope and functionality of the system were labelled `OOD` (out-of-domain). To exemplify the labelling process, Figure 2 shows an actual user utterance, together with its corresponding parse and transcript, and the labels for each of the concepts.

| | |
|---|---|
| **System:** | `Where are you leaving from?` |
| **User (transcript):** | *#noise# from Paris to London and then on to Toronto #noise#* |
| **Decoded hypothesis:** | `from Paris to London on then on to go on to` |
| **Parse:** | [DepartLocation] `from Paris` [ArriveLocation] `to London` [Interjection] `then` [Resume] `go on` |
| **Labelling:** | DepartLocation/`OK` ArriveLocation/`OK` Interjection/`OK` Resume/`RBAD` |

Figure 2: Sample user utterance, with transcript, parse and concept labelling

Next, aggregate binary whole-utterance labels were generated automatically: any combination of `BAD`-type labels was labelled `BAD`, while utterances containing only `OK` labels were labelled `OK`. The labelling was initially performed at the concept level, as we were originally also interested in the problem of concept-level confidence annotation. Surprisingly, simple statistics on the concept labels indicated that only a small number of utterances - 6% in OctNov99 and only 2% in JuneJuly01 corpus - contained combinations of `OK` and `BAD`-type concepts (the utterance from Figure 2 is actually one such utterance). We believe that this somewhat surprising result can be explained to a large extent by the fact that the average utterance length in terms of concepts was relatively low.

Given this very small percentage of mix-labelled utterances, we discarded these utterances from the training corpus for the purposes of developing a binary utterance-level confidence annotator, and we abandoned for the moment the idea of constructing a concept-level confidence annotator.

#### 3.1.2 Feature identification

Identifying relevant features is paramount for the success of any classifier. Confidence annotators for speech recognizers have traditionally been built upon features provided by the speech recognition layer [12, 13, 14, 15]. However, we argue that in the context of a fully functioning dialog system there are at least two other orthogonal knowledge sources which carry valuable information for the confidence annotation process: the parser and the dialog manager. Intuitively, a very good parse result is not likely to be obtained in the context of misunderstandings. Also, dialog-level information (e.g. the dialog state, the match between the parse and the list of expected concepts, etc) can be very informative for the process of confidence annotation. We therefore selected and integrated useful features from all these three different levels of the dialog system.

Based on the previous example (see Figure 2), we now describe in detail each of the features that were identified and used in the classification process. It is important to note that all these features are directly computable both at runtime (as they will later be needed by the running confidence annotator), and also from the information stored in the system's logs, as they are used in the training and testing process. Furthermore, although collected from our particular implementation of the Communicator system, these features are by-and-large task- and system- independent.

### A. Speech Recognition-Level Features

We limited ourselves to the use of two easily computable features from the speech recognition level:

- **Word Number**: the number of word tokens in an utterance (as decoded by the SPHINX recognition engine) - 11 in the example above. Although this feature might not seem very informative for the confidence annotation process by itself, its correlation to other features (like **Concept Number** from the parsing level) could be exploited by an appropriate classifier: an utterance containing a lot of words and very few or no concepts, is very likely a misunderstanding.

- **Unconfident Percentage**: the percentage of word tokens tagged by the decoded with an unconfident marker. The intuition is that a high unconfident percentage is often an indication of unconfident input. In the example in Figure 2, there are no unconfident tagged words, so the unconfident percentage is 0.

Typically, confidence annotators for speech recognition use a multitude of other features derived from the speech recognition layer. The choice of using only these 2 decoder-level features for the utterance-level confidence annotator is justified though in this setting by the fact that the second feature (the percentage of unconfident words), already captures and integrates all the useful sources of information from the recognition layer.

### B. Parsing-Level Features

Six relevant features were identified and used from the parsing level:

- **Uncovered Percentage**: the percentage of the word tokens in the utterance that are not covered by the parse (the words covered by the parse are printed in bold typeface in Figure 2). Similar to the **Unconfident Percentage**, a high value for this feature most likely signals an unreliable input. In the example illustrated in Figure 2, there are 4 words uncovered by the parse: *on*, a second *on*, *to* and the last *to* and therefore **Uncovered Percentage** = 4/11.

- **Fragment Transitions**: the number of transitions between parsed and unparsed fragments in the utterance. This feature measures the fragmentation degree of the parse, with high values indicating possible misunderstandings. In the sample utterance above, **Fragment Transitions** = 5.

- **Gap Number**: the number of unparsed fragments in the sentence. High values for this feature indicate high parse fragmentation, and thus an increased likelihood of misunderstanding. In the example above, we have 3 gaps: *on*, *on to* and *to*, so **Gap Number** = 3.

- **Concept Number**: the number of concepts in the parsing result. There are 4 concepts in the sample utterance above: [DepartureLocation], [ArrivalLocation] , [Interjection] and [Resume]. As previously mentioned, the possible correlation between this feature and the **Word Number** feature could be exploited by an appropriate classifier to signal misunderstandings.

- **Bigram Score**: the score assigned to the parse by a bigram language model constructed using grammar slots as tokens. This type of feature, has also been successfully used for recognizer hypothesis rescoring [24]. The feature captures the coherence of the parse, a high score meaning high coherence, and therefore a reduced likelihood of misunderstandings.

- **Garble**: the binary indicator previously used in the Communicator system to signal misunderstandings. This flag is set by the post-processing module HELIOS, which employed a heuristic rule based on low parse coverage and high fragmentation to detect misunderstandings. For the sample utterance above, the **Garble** flag was not raised, and therefore **Garble** = 0.

### C. Dialog-Management Level Features

From the dialog management level, we used the following 4 features:

- **Dialog State**: this feature indicates which state the Dialog Manager is currently in. As the dialog manager in the CMU Communicator is not state-based (see the AGENDA architecture [3]), the state information is generated on the basis of a manual clustering of system prompts into categories that entrain similar inputs from the user. **Dialog State** is a nominal feature, having one of 17 possible values. In the example above, the state is *query_depart_location*.

- **State Duration**: this feature captures the number of consecutive turns for which the system has remained in the current state. High values indicate an increased likelihood of repeated misunderstandings.

- **Turn Number**: the number of turns from the start of the conversation. Under normal conditions (no or very little misunderstandings), there should be a correlation between the turn number and the **Dialog State** feature, which can be captured and exploited by an appropriate classifier.

- **Expected Concepts**: this feature indicates whether or not the concepts in the parsing result correspond to the current expectation of the dialog manager. Expectation is defined in the CMU Communicator, as a three-fold nominal feature: *expected*, *accepted* and *unexpected*. A concept (grammar slot) is *expected* if it is consumed by the handler which is currently in focus on the agenda [3], and it would be considered *accepted* if it is consumed by a handler which is not the current focus. If no handler binds to a particular concept in the parse, the concept is considered *unexpected*. For instance, for the utterance in Figure 2, [DepartureLocation] is *expected*, [ArrivalLocation] is *accepted*, and [Interjection] and [Resume] are *unexpected*.

### 3.1.3 Automated Classification Techniques

Several different machine learning classification techniques were explored, in an attempt to empirically establish which was the most appropriate for the task at hand. Below, we comment briefly on each of these techniques and the rationale for choosing each of them as possible solutions.

**Bayesian Networks** [25] provide a probabilistic framework for inference and decision-making, which is very appropriate given the nature of the confidence annotation task. This approach can perform a quantitative assessment of the evidence coming from multiple sources, and supporting several hypotheses. Initially, a structure linking each feature to the classification outcome was used. After some experimentation we discovered that limiting the structure to a smaller number of features produced better results: the explanation lies in the fact that a network with fewer nodes is less prone to overfitting, and that the feature set chosen exhibits some redundancies.

**Boosting** (AdaBoost.M2) [26] is a voting technique that combines a set of weak learners (the performance of each individual learner needs to be slightly better than random) and iteratively boosts their performance by changing the distribution over the training examples to focus the learners on the hard instances. This technique is appealing in our setting, since we can use individual feature-based predictors as weak learners. The boosting process iteratively chooses the best performing features, and assigns weights to each feature according to its predictive power, thus eliminating the problem of redundant information in the feature set.

Another widely used classification technique we decided to explore was **Decision Trees** [27]. In this approach classification is performed by dividing the feature space into sub-spaces, and ultimately identifying each sub-space with a corresponding majority class. The partitioning process is implemented by iteratively choosing the next best feature, based on information gain. As with AdaBoost, redundancies in the feature set do not pose problems here.

Next, we turned to **Artificial Neural Networks** (ANN) [29]. This type of classifier is able to learn complex functions with continuous valued output, and is generally robust to noise in the training set. We used a typical 3 layer feed-forward architecture trained using the back-propagation algorithm.

**Support Vector Machines** [30] have received a great deal of attention in recent years. It has been shown that on some domains the performance of this approach matches and sometimes exceeds that of traditional approaches to classification like decision trees and neural networks. SVMs essentially work by mapping the samples into a highly dimensional space, and looking for an optimal linear separator in that space [30].

Finally, we also constructed a **Naïve Bayes** classifier [28]. Although this classifier makes the assumption of independence between features (which is not really the case in our setting), it is very simple, easy to implement and has been used with a lot of success in text-based domains.

### 3.1.4 Experiments and Results

We now describe the experiments performed and comment on the results that were obtained using each of the previously mentioned classifiers.

The first set of experiments were based on the OctNov99 corpus. In order to be able to reliably compare the various techniques used, all the experiments involved a 10-fold cross-validation process on the specified data-set. For the moment, the performance of each classifier is characterized by the mean and variance of the error rates in this process. To build a more accurate picture of the classification performance, we also report the false-positive and false-negative rates. Later on, in Section 4, we take into account the impact of these various types of errors on the dialog system, and refine the performance metric.

Two different performance baselines are used: the first one is given by the error rate of a naive classifier that would always choose the majority class - in our case `BAD`; this majority baseline is at 32.8% (the same as the ratio of `OK` utterances in the corpus). A second baseline is given by the performance of the previous rule-based misunderstanding detector operating in HELIOS - the **Garble** indicator. This second baseline error rate is significantly lower - 25.3%.

### A. Performance of individual features

Prior to the actual experiments with the classifiers, we evaluated how well each feature is able to predict the target labels by itself. Individual histogram-based predictors were constructed for each feature, and evaluated in a 10-fold cross-validation process on both data-sets. The results are presented in Table 2.

| **Feature** | **Mean Error Rate (%)** | **F/P Rate** (%) | **F/N Rate** (%) |
|---|---|---|---|
| **Majority Baseline** | **32.84** | **-** | **-** |
| Uncovered Percentage | 19.93 | 17.60 | 2.33 |
| Expected Concepts | 20.97 | 12.24 | 8.73 |
| Gap Number | 23.01 | 14.51 | 8.51 |
| Bigram Score | 23.14 | 15.80 | 7.34 |
| Garble | 25.32 | 25.30 | 0.02 |
| Slot Number | 26.69 | 25.52 | 0.18 |
| Unconfident Percentage | 27.34 | 26.18 | 1.16 |
| Dialog State | 31.03 | 25.82 | 5.21 |
| Word Number | 32.33 | 32.07 | 0.26 |
| Fragment Number | 32.73 | 27.78 | 4.95 |
| State Duration | 32.84 | 32.02 | 0.81 |
| Turn Number | 33.14 | 32.40 | 0.75 |

Table 2: Performance of individual feature-based classifiers

As Table 2 illustrates, the most useful features are **Uncovered Percentage**, **Expected Concepts**, **Gap Number** and **Bigram Score**; they each perform better than the **Garble** baseline. The worst performing features are **Word Number**, **Fragment Number**, **State Duration** and **Turn Number**. This is somewhat expected, as these features intuitively don't carry a lot of information just by themselves. **Garble**, the rule-based classifier previously used in the system to detect misunderstandings ranks fifth. It has however the lowest false-negatives rate, intuitively a very desirable property for a misunderstanding detector.

### B. Classifier performance

Table 3 illustrates the results of the 6 classifiers on the OctNov99 dataset. Judged from a classification error rate perspective, **AdaBoost** obtains the best result (16.59%), equivalent with a 34.4% relative reduction in error rate from the previously used **Garble** rule, and 49.5% from the majority baseline. The runner-ups are the **Decision Tree** and the **Bayesian Network** classifiers. A t-test indicated however that the differences between the mean error rates of the top 5 classifiers are not statistically significant at the 0.05 level of confidence.

| Classifier | Mean Error Rate (%) | F/P Rate (%) | F/N Rate (%) |
|---|---|---|---|
| **Garble Baseline** | **25.32** | **25.30** | **0.02** |
| AdaBoost | 16.59 | 11.43 | 5.16 |
| Decision Tree | 17.32 | 11.82 | 5.49 |
| Bayesian Net | 17.82 | 9.41 | 8.42 |
| SVM | 18.40 | 15.01 | 3.39 |
| Neural Net | 18.90 | 15.08 | 3.82 |
| Naïve Bayes | 21.65 | 14.24 | 7.41 |

Table 3: Classification performance

The Bayesian Network classifier has the lowest false-negative error rate (8.42%), but this is still significantly higher than that of our previously used Garble heuristic rule. The Naïve Bayes classifier performs the worst. We suspect this is due to the assumption of independence between features made by this classifier, assumption clearly violated in this case.

In conclusion, the experiments performed indicate that the problem of detecting misunderstandings can be approached successfully by pulling together various sources of knowledge from different levels in the dialog system in a machine learning approach. Although the resulting classification error rates are far from optimal (zero), the reductions are significant compared to both the majority baseline (about 50% relative reduction) and to our **Garble** heuristic baseline (about 34% relative reduction). The consistency of the results and the absence of a single best performing classifier seem to confirm that indeed we have reached optimal performance given the set of features we have chosen. In the future, we intend to concentrate on identifying more features, with the hope of driving the error rates even lower.

### 3.2 Continuous-Score Utterance Level Confidence Annotation

Next, we address the second flavor of the confidence annotation problem. This time the goal is to obtain a continuous, probability-like score which characterizes the degree of belief in a certain perceived utterance, rather than a mere accept / reject decision.

Each of the classifiers evaluated in the previous section can provide such a continuous score; most of them are discriminant in nature and actually work by computing a real-valued classification margin and then applying a threshold to make a binary decision. A direct, but, as we shall shortly see, not very performant approach would thus be to simply scale the values of the margins so as to bring them into the unity interval, and then use that value as a probability score. The experiments presented in the sequel will confirm that, indeed, the use of margins from discriminant classifiers as probability-like scores is not very reliable, and a density estimation method is preferable in this setting.

We performed an empirical comparison between two qualitatively different techniques:

AdaBoost, which is a discriminant classifier on one side, and a logistic regression model which is a density estimator on the other side. Discriminant classification techniques construct a boundary in the feature space which separates as well as possible the different classes. By contrast, density estimation techniques construct statistical models which try to explain or account for the data in the training set. By avoiding the construction of a full-blown model for the data, discriminant techniques can sometimes be more straightforward, and are more appropriate when we are simply interested in distinguishing between classes, and not in obtaining a degree of belief in that decision. However, once we become interested in obtaining this degree of belief, the performance of these classifiers will be lacking as compared to density estimation models.

In logistic regression [31], an explanatory model for the data is constructed; the model belongs to the generalized linear models family, and expresses the logit of the target value as a linear combination of features. As nominal features cannot be naturally included in a logistic regression model, we discarded the Dialog State feature, and constructed this model using the remaining 11 features. The same 10-fold cross-validation process was used to build and evaluate this model based on the OctNov99 corpus. The experiments indicated a mean classification error rate of 17.14%, very similar to the performance of the six classifiers previously discussed.

In this setting, evaluating performance via a binary classification error rate (the hard error rate) is no longer adequate. Instead, one should use an entropy-like soft error metric: the average log posterior of the correct label, as defined by equation (1) (where P is the set of positive instances and N is the set of negative ones). For every instance, the classifier will output a degree of belief of correct understanding p in [0,1]. A binary acceptance decision typically corresponds to values of $p > 0.5$. The soft metric averages the log-likelihood of correct decisions (i.e. if the utterance was indeed correct we take $\log(p)$, otherwise $\log(1-p)$), taking thus into account the "magnitude" of the error. The soft error metric can also be viewed as the cross-entropy between the constructed model and the "real world model" (high values are better).

$$SE = \sum_{i \in P} \log(p_i) + \sum_{i \in N} \log(1 - p_i), \tag{1}$$

Although on the hard error metric, the results of the logistic regression model were very similar to those of AdaBoost, this situation changes dramatically on the soft error metric. The soft error metric for the AdaBoost classifier was computed by linearly scaling the margin of each instance to produce a probability-like score. Figure 3 illustrates the AdaBoost soft-error, as a function of the number of boosting stages. This curve indicates that strong overfitting takes place (in the soft-error sense). After a small number of boosting stages the soft-error performance of AdaBoost reaches a maximum of -0.6614, point from which it continuously declines. This overfitting nature of the curve has a simple explanation: as the number of boosting stages increases, AdaBoost continuously updates the classification boundary so as to maximize the margins [26]. Therefore, after a large number of boosting stages, the margins for most instances tend to be large. This implies that the degree of confidence in the results outputted by AdaBoost is artificially increased, and therefore whenever an error is committed, a high penalty in terms of soft-error will be incurred.

Figure 3 also illustrates the soft-error rate for the logistic regression model: -0.579. Compared with the maximum of -0.6614 reached by AdaBoost, this is a significantly better result. These experiments have therefore confirmed that, if a degree of belief in correct understanding is needed (for instance for later integration in a Bayesian decision framework for clarification and confirmation at the dialog management level), density estimation models are to be preferred to discriminant techniques.

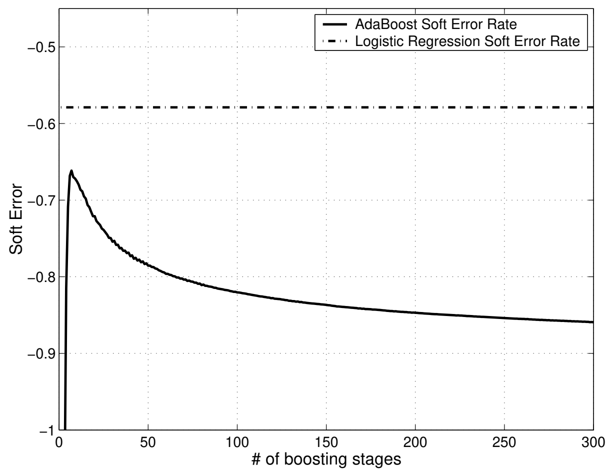


Figure 3: Soft error rate for logistic regression and AdaBoost (as a function of # of boosting stages)

We have so far illustrated the development of an utterance-level confidence annotator. Experiments have shown that multiple task-independent features from different levels in the dialog system can be identified, and integrated in a misunderstanding detector using various machine-learning techniques. For the binary decision flavor of the problem, we have shown that several of these techniques can all lead to significant improvements in classification error rate, and no preferred solution could be clearly identified. Moreover, if interested in obtaining a probability-score reflecting the belief in correct perception of a certain utterance, a density estimator model is to be favored to a discriminant technique.

# 4 Modelling the Costs of Misunderstandings

We now turn to the second part of the problem we have stated: that of modelling the costs of the various types of errors committed by the confidence annotator, with a view to assessing their impact on dialog, and to fine-tuning the annotator's performance.

We start by motivating the need for this type of cost assessment in Section 4.1, and subsequently we describe the approach taken to solving this problem (Section 4.2). Section 4.3 describes in detail the experiments performed on developing several cost models based on various dialog performance metrics, and in Section 4.4 we show how the models constructed can be used to fine-tune the confidence annotator. Finally, in Section 4.5 we take a closer look at the factors involved in these experiments, in an effort to gain a better understanding of the somewhat surprising results we obtain.

## 4.1 Motivation for error cost analysis

An issue we have not addressed so far in the development of the utterance level confidence annotator is that of modelling the costs of the various types of errors that it commits. When training a classifier, as in the experiments previously described, we are typically minimizing the total classification error rate. We argue that, given the role played by the classifier - a detector for misunderstandings in dialog, and given that different types of errors are likely to have a different impact on dialog performance, classification error rate is not an appropriate metric for evaluating the confidence annotator.

The classification error rate can simply be decomposed as the sum of the false-positives and the false-negatives that the annotator commits. Therefore, in striving to minimize classification error rate, we are minimizing this sum, and thus implicitly making the assumption that the cost of these two types of errors are the same. However, intuition tells us that this assumption is violated in most spoken dialog systems. On a false-positive error, the system will accept, and possibly use invalid information. This will most likely require correction later on, which will slow down the progress of the dialog. By contrast, on a false-negative error, a system will usually communicate the (false) misunderstanding to the user and ask the user to repeat the last utterance. We believe that a false-positive (an undetected misunderstanding) would typically cause more trouble later on in the dialog, and thus we would expect its cost to be higher: better safe than sorry.

The trade-off point between the two types of errors is nevertheless not evident, and is highly system-dependent. For instance, the cost of a false-positive is likely to be higher in a system that uses an undo mechanism than in a system that allows the users to easily overwrite incorrectly captured concepts. Moreover, costs can vary from one domain to another, or, even within the same system, from one point in the dialog to another. A false-positive error on a command issued to the system should be more costly than when the user is merely communication some piece of information, which could presumably be easily overwritten.

In the following sections we will describe a data-driven approach which allows us to quantitatively assess the costs of the various types of errors that the confidence annotator commits. Once these costs are known, we can use them to optimize the annotator accordingly: instead of simply minimizing the sum of false-negatives and false-positives (i.e. the classification error rate), we can minimize their weighted sum, taking the costs of these errors into account.

## 4.2 Approach

We propose a data-driven approach to quantitatively assess the costs of the errors committed by the confidence annotator. In brief, the idea is to construct a statistical regression model which links the counts of the errors in each session to the performance of the dialog system in that session. The coefficients obtained in the regression will thus indicate the impact of each type of error on the performance of the dialog system.

As a first step, we need to identify a suitable dialog performance metric. This performance metric will play the role of target variable for the regression model, and needs to be easily computable from the training corpus. Several dialog performance metrics have been proposed and used in the literature. More complex evaluations, taking into account a variety of factors have been performed in the PARADISE framework [32].

In our work, both objective and subjective performance evaluation metrics were considered. For instance, a widely used performance metric for spoken dialog systems is user satisfaction. We therefore collected user satisfaction scores on a 5-point Likert scale for the JuneJuly01 corpus. However, experiments have confirmed this metric to be very subjective, inconsistent across users and unreliable (see more details in Section 4.3.2). Most of our attention therefore became focused on objective metrics such as dialog efficiency (the rate at which correct concepts are transferred from the user to the system), and binary task completion. All these dialog performance metrics, together with the cost models created using them, are discussed in detail in the next section.

Once a performance metric $P$ to optimize for is chosen, the next step is to create a statistical regression model which uses this metric as a response variable, and the counts of the various types of errors as predictor variables, as in equation 2.

$$P = f(FP, FN) \tag{2}$$

Obtaining a good fit gives us a robust quantitative assessment of the (negative) contribution of these different errors to the performance of the system. If the model is a linear one, the obtained coefficients of the predictor variables can be directly interpreted as the costs of these various types of errors (see equation 3):

$$P = Cost_{FP} \cdot FP + Cost_{FN} \cdot FN \tag{3}$$

Subsequently, the model can be used used to optimally[1] fine-tune the classifier, by changing the classification threshold to minimize for $f$ instead of the sum of the errors; this idea is discussed in further detail in Section 4.4.

It is important to notice that with this approach we have separated the models for the costs of the errors from the confidence annotator itself. An alternative would have been to perform the whole optimization in one step, by training the classifier with different target labels, which would have been derived from (and would thus reflect) the dialog performance on each utterance. There are several advantages however to decoupling the two: first, it allows us to fine-tune a previously developed confidence annotator to any given spoken dialog system, without retraining it. Second, as previously mentioned, if we use a linear regression model we can obtain a direct quantitative assessment of the costs (i.e. the coefficients of regression). More importantly, since the regression model is constructed using whole sessions as data-points, we can target global performance metrics, and thus capture long-range (across an entire session) effects of the confidence errors rather than effects within any single utterance.

Conversely, one drawback of the proposed approach is that it makes the assumption that the cost of errors is constant throughout the dialog. This assumption could nevertheless be relaxed, by introducing extra predictor variables, in the presence of a sufficiently large training corpus. For instance, one could differentiate costs based on dialog state (errors committed whilst in different dialog states have different costs), by introducing a predictor variable for each type of error and state (see equation 4, where $S$ represents the set of all states:

$$P = \sum_{s \in S} (Cost_{FP_s} \cdot FP_s + Cost_{FN_s} \cdot FN_s) \tag{4}$$

This will however quickly lead to a severe data scarcity problem, and unless the number of states is very small or we benefit of a very large corpus, the approach quickly becomes infeasible.

Finally, before moving on to present the actual experiments and the results that were obtained, we briefly comment on similar work in the literature. Smith and Hipp [7] propose the use of dialog work analysis to determine the trade-off point between these types of errors. Compared to their approach, ours is entirely data-driven. Our solution also bears

[1] where optimality is defined in terms of the chosen metric

resemblance to the PARADISE framework [32], in which regression models are used to evaluate dialog performance. Our work is different though in that it is targeted at assessing the costs of several precise types of errors with the final goal of optimizing the performance of the confidence annotator for a particular spoken dialog system. To our knowledge, this is the first empirical investigation of the costs of misunderstanding errors in spoken dialog systems.

## 4.3 Cost Modelling Experiments

We now describe the experiments performed and the results obtained in constructing the cost models. In Section 4.3.1 we present the incremental development of three successively refined models using dialog efficiency as the targeted response variable. Subsequently, in Section 4.3.2 we describe two additional models constructed using task completion and user satisfaction as target performance metrics.

The cost modelling experiments were performed based on the JuneJuly01 corpus, described previously in Section 3.1.1. Additional statistics for this corpus, relevant to this context, are presented in Table 5.

### 4.3.1 Cost Models targeting Dialog Efficiency

One good indicator of dialog performance is dialog efficiency, measured as the rate at which the system obtains accurate information from the user. This is an objective metric, which can easily be computed for each session from the system logs, and it captures to a large extent the goodness-of-the-dialog, as the timely completion of a given task typically requires the system to correctly acquire various concepts from the user. Three successive models were developed using efficiency as the targeted dialog performance metric; they are described in depth in the following subsections.

**Model 1: CTC = FP + FN + TN**

The response variable used to assess dialog efficiency for this first model was the number of ***correctly transferred concepts per turn*** (CTC in the sequel). CTC is computed by counting the number of concepts correctly transferred from the user to the system throughout a whole session, and dividing that by the number of turns in that session. A concept is considered correctly transferred when: (1) it has an OK concept label, (2) it is actually used by the Dialog Manager and (3) the respective utterance is accepted by the system (i.e. the confidence annotator does not reject it). The second condition is imposed as some of the concepts appearing in the semantic grammar in the Communicator system are not used by the Dialog Manager (see the example in Figure 4). The list of concepts that the Dialog Manager looks for at a particular point is dynamically determined at run-time, based on the list of unblocked handlers currently in the agenda. This information is also logged. The veracity of the other two conditions is also easily determined off-line, from the available logs.

To exemplify, in the fictive utterance presented in Figure 4, there is only one correctly transferred concept - [DepartLocation]. Although the concept label for [I_Want] is also OK, this concept is not used by the system, and thus does not satisfy the second condition. Also, note that if the confidence annotator's decision had been to reject this utterance, then CTC would be 0, as no transfer of information from the user to the system would have occurred.

| | |
|---|---|
| **User says:** | *I want to fly from Pittsburgh to Boston* |
| **System recognizes:** | `I want to fly from Pittsburgh to Austin` |
| **Concepts:** | [I_want/`OK`] [DepartLocation/`OK`] [ArriveLocation/`RBAD`] |
| **Annotator decision:** | Accept |

Figure 4: Fictive utterance, illustrating concept labelling and efficiency metric computation

The predictor variables for this first cost model are the rates of ***false-positives*** (FP), ***false-negatives*** (FN) and ***true-negatives*** (TN) committed by the utterance-level confidence annotator. The inclusion of the true-negatives rate (i.e. the correctly detected misunderstandings) as a predictor in this model is justified, as the models will be later used to fine-tune the annotator by changing the classification threshold. This will typically break the balance not only between the numbers of false-positive and false-negative errors, but also true-negative errors, so therefore they need to be accounted for by the model.

A linear regression model was constructed, and the results are illustrated in Table 4. The $R^2$ value of 0.82 indicates a good fit. The robustness of the model was also verified in a 10-fold cross-validation experiment. The means of the $R^2$ values in the 10 runs on the training and testing set are shown in Table 4.

| **Model** | $R^2$ on entire data-set | $R^2$ on training set | $R^2$ on testing set |
|---|---|---|---|
| CTC = FP+FN+TN | 0.8160 | 0.8169 | 0.7336 |
| CTC-ITC = FP+FN+TN | 0.8650 | 0.8657 | 0.7866 |
| CTC-ITC = REC+FP+FN+TN | 0.8910 | 0.8912 | 0.8325 |
| CTC-ITC = REC+FPNC+FPC+FN+TN | 0.9436 | 0.9439 | 0.9014 |

Table 4: Cost models targeting dialog efficiency

**Model 2: CTC-ITC = REC + FP + FN + TN**

The first model was refined by changing the response variable to also account for ***incorrectly transferred concepts*** (ITC in the sequel). A concept is incorrectly transferred from the user to the system when: (1) it has a `BAD`-type label, (2) it is used by the Dialog Manager and (3) the confidence annotator accepts the utterance as correct. For instance, in the utterance presented in Figure 4, there is one incorrectly transferred concept: the arrival location [ArriveLocation].

Optimally, we want not only to maximize the correctly transferred concepts, but also to minimize the number of incorrectly transferred ones. This being said, a more encompassing efficiency performance metric is CTC-ITC. Experiments were repeated using this term as the response variable, and, as Table 4 illustrates, this refinement improved the fit, leading to a new $R^2 = 0.86$.

A second refinement to the model addresses the verbosity of the user. In the context of a mixed-initiative spoken dialog system like the CMU Communicator, a more verbose user could achieve the same task with a smaller number of turns, and thus a higher CTC than a terse user. To account for this effect, we added another predictor variable: the number of

***relevantly expressed concepts*** per turn (REC in the sequel). A relevantly expressed concept is a concept that binds to one of the active handlers on the dialog manager agenda (i.e. concepts which are taken into account by the dialog manager), regardless of its label and of the decision made by the confidence annotator on that particular utterance. To exemplify, in the utterance in Figure 4, there are 2 relevantly expressed concepts: [DepartLocation] and [ArriveLocation].

Adding this new predictor variable improved the fit even more, leading to an $R^2 = 0.89$ (see Table 4).

**Model 3: CTC-ITC=REC+FPNC+FPC+FN+TN**

An inspection of the coefficients computed in the last linear regression model showed that the costs for the false-positives and the false-negatives were very similar: -1.46 and -1.44 respectively. This contradicts the intuition that the cost should be higher for false-positives, and a careful analysis of this result led to an additional refinement of the model.

An important observation is that there are two conceptually different types of false-positive errors in the CMU Communicator dialog system. If the utterance contains relevantly expressed concepts, and the confidence annotator commits a false-positive, the system will accept and act based on invalid information (e.g. using Austin instead of Boston as the arrival city in the previous example). However, if there are no relevantly expressed concepts, even if the system wrongly accepts the utterance, it will find no useful information in it (we have a non-understanding); ultimately the system will inform the user that it misunderstood, essentially acting the same as on a true-negative. We deem the first type of false-positive error a ***False-Positive with Concepts*** (FPC), and the second type a ***False-Positive with No Concepts*** (FPNC).

The impact of these two types of false-positives on the dialog is clearly different, and the intuition is that the cost for false-positives with concepts is the one that should be significantly higher. Therefore, in the third model, we replaced the FP predictor variable by FPC and FPNC. This model provides an even better fit: $R^2 = 0.94$ (see Table 4.)

The coefficients obtained lead to the following regression formula (5) which captures the impact of each of the factors on dialog performance:

$$Efficiency = 0.42 + 0.63 \cdot REC - 0.48 \cdot FPNC - 2.12 \cdot FPC - 1.33 \cdot FN - 0.56 \cdot TN \quad (5)$$

| Dataset Statistics | Total | Mean per Dialog | Std.Dev. per Dialog |
|---|---|---|---|
| Number of dialogs | 134 | - | - |
| Number of utterances | 2561 | 19.11 | 9.34 |
| #CTC | 1983 | 14.80 | 7.64 |
| #ITC | 166 | 1.24 | 1.61 |
| #REC | 2373 | 17.71 | 9.25 |
| CTC/Turn | - | 0.77 | 0.23 |
| (CTC-ITC)/Turn | - | 0.71 | 0.28 |

Table 5: Additional cost-model relevant statistics for the JuneJuly01 corpus

| | Coefficient | 95% Confidence interv. |
|---|---|---|
| Constant term | 0.4188 | [0.3075 - 0.5302] |
| REC | 0.6254 | [0.5269 - 0.7239] |
| FPNC | -0.4820 | [-0.6693 - -0.2707] |
| FPC | -2.1222 | [-2.2894 - -1.9550] |
| FN | -1.3302 | [-1.5429 - -1.1175] |
| TN | -0.5588 | [-0.7025 - -0.4151] |

Table 6: Estimated cost coefficients (and 95% confidence intervals)

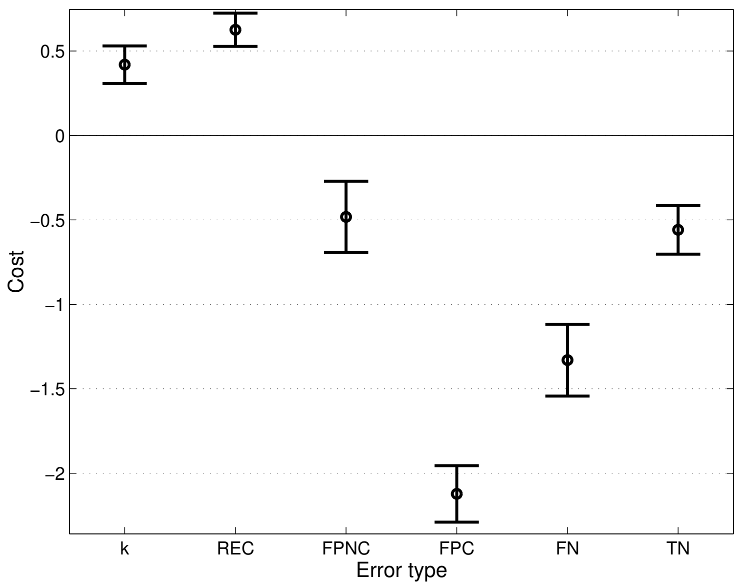


Figure 5: Plot of estimated cost coefficients (and 95% confidence intervals)

Table 6 and Figure 5 illustrate the coefficients from this third model, together with their 95% confidence intervals obtained in the cross-validation process. These results confirm our previously stated intuition: the cost of a false-positive with concepts is the greatest, being 1.6 times larger than that of a false-negative error. The cost of a false-positive without concepts is very similar to that of a true negative, and the REC factor bears a positive coefficient, as it actually increases dialog efficiency.

#### 4.3.2 Other Cost Models: Completion Rate and User Satisfaction

**A. Models based on task completion**

While a model of net concept transmission efficiency is of immediate interest in determining how to use a confidence annotator, we can also consider other response variables that appear to be correlated with good dialogs, for instance *task completion*.

Task completion belongs to the family of objective dialog performance metrics, and can be automatically computed in the presence of an attribute-value matrix [32]. The dynamic nature of the constraints assembled by the dialog manager in the Communicator system precludes however the direct use of this technique, since the number of attributes is not apriori known. We therefore turned to a simplified *binary definition of task completion*: a human annotator labelled each of the sessions as completed or not, based on the information available from the logs. A shortfall of this metric is that it does not capture correctness: in some sessions, although completed, due to various misunderstandings the system had failed to provide the appropriate information to the user. We therefore further defined *correct completion*, by adding the constraint that the system responds with the correct information required by the user (as the transcripts indicate).

We attempted to construct models based on both these metrics. Since both completion and correct completion are binary variables, we shifted from the use of the linear regression models (no longer appropriate) to logistic regression models. Evaluation was performed in terms of classification error rate on the training set. The results however did not indicate a good fit, as illustrated in Table 7. This was not very surprising, given the coarse granularity definition of completion and correct completion used, and given that factors other than utterance rejection are also likely to affect task completion. We believe that a metric like completion could in principle be used successfully in this setting, if it can be computed on a finer granularity level, for instance based on the attribute-value matrix as in [32].

| **Metric** | **Baseline** | **Error on training set** |
|---|---|---|
| Completion | 43.81% | 23.81% |
| Correct Completion | 40.00% | 25.71% |

Table 7: Performance of completion-based models (classification error rates)

**B. Models based on user satisfaction**

Another dialog performance metric that could be used as a response variable for our cost models is user satisfaction. Following the PARADISE framework [32], we used completion and accuracy as the predictor variables. Since we were interested in the individual contributions of the various types of errors that the confidence annotator commits, accuracy was decomposed into the FP, FN and TN factors.

Unfortunately, our corpus contained user satisfaction scores for only 35 dialogs. The fit for the model constructed using these data-points - $R^2 = 0.61$ - is comparable with other results reported in the literature [32], but is significantly lower than the $R^2$ values obtained using dialog efficiency as a response variable. We believe that this is mostly due to the subjectivity of the user satisfaction metric (different users have different expectations from the system, and different definitions of satisfaction); the limited number of available data-points might also have played an important role in this result.

### 4.4 Fine-tuning the confidence annotator

We now illustrate how to make use of the cost models developed previously, in an effort to find the optimal trade-off point between the various types of errors committed by the confidence annotator.

In order to make a hard, binary decision, most classifiers compare the output of the classification process (typically a continuous-valued score) with a certain threshold. In AdaBoost [26] for instance, the combined vote of the weak learners is compared with 0: if the vote is greater than zero, the instance is labelled as a positive, otherwise as a negative. By changing the value of this threshold we can bias the classifier towards more false-positive or more false-negative errors. Figure 6 illustrates the error rates for the different types of errors (FPNC, FPC, FN) that the logistic regression confidence annotator makes, as a function of the classification threshold (the classifier operating characteristic). The classification error rate (FPNC + FPC + FN) is also illustrated, and, as expected, it is minimized for a 0.5 value of the threshold (the default threshold for a logistic regression classifier).

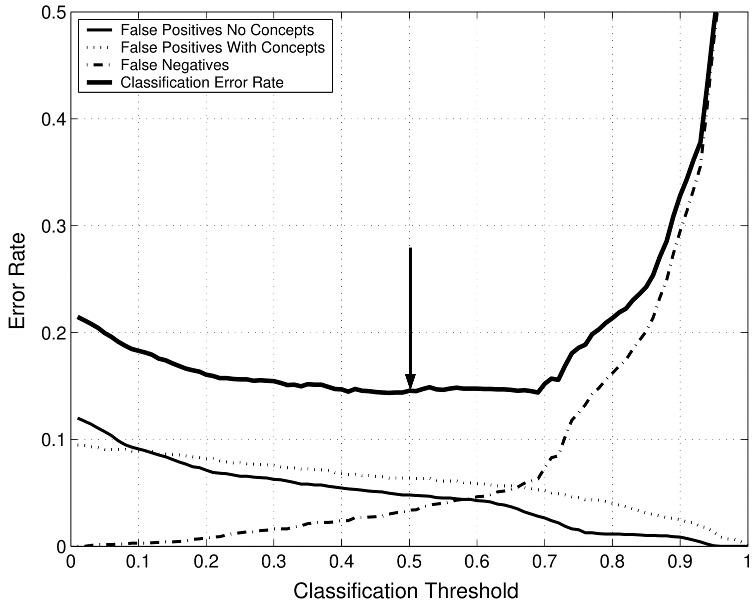


Figure 6: Different types of errors as a function of the classification threshold for the logistic regression confidence annotator

In order to determine the optimal tradeoff point between all these types of errors - where optimal means taking costs into account - we need to identify the threshold value which maximizes the regression expression, and therefore implicitly the response variable, i.e. the selected dialog performance metric.

Using the third model for dialog efficiency, we have:

$$Efficiency = 0.42 + 0.63 \cdot REC - 0.48 \cdot FPNC - 2.12 \cdot FPC - 1.33 \cdot FN - 0.56 \cdot TN$$

Since the REC factor (which models the user's verbosity) is independent of the chosen threshold, and since the constant factor does not affect the location of the maximum, maximizing efficiency is equivalent to minimizing the following cost:

$$TotalCost = 0.48 \cdot FPNC + 2.12 \cdot FPC + 1.33 \cdot FN + 0.56 \cdot TN$$

This total cost, as a function of the chosen threshold, is plotted in Figure 7. No minimum could be clearly identified. This is a surprising, somewhat counterintuitive, and very

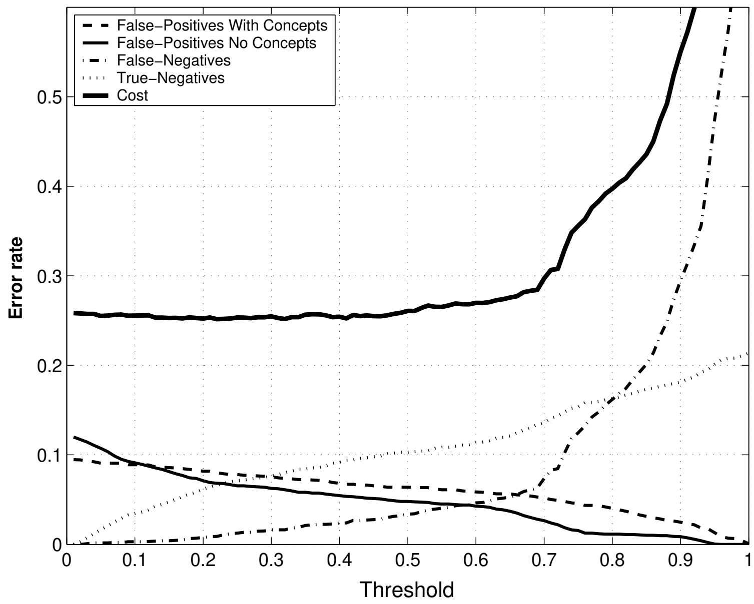


Figure 7: Cost as a function of confidence annotation threshold

interesting result. The fact that the cost function is almost constant across a wide range [0-0.5] of the operating characteristic indicates that, to a large extent, the efficiency of the dialog stays about the same (at least in terms of the metric we have chosen to investigate), regardless of the ratio of false-positive and false-negative errors that the system makes. Even when the threshold is set to zero, which is equivalent to completely eliminating the utterance level confidence annotator, the degradation in efficiency as measured by CTC-ITC would be insignificant. A very similar result was obtained for the AdaBoost-based confidence annotator.

### 4.5 Further Analysis

In an effort to gain a better understanding of this unexpected result, we performed several additional experiments, described below.

We started by reanalyzing the appropriateness of CTC - ITC as a dialog performance metric. An analysis of the distribution of this variable across the sessions in the JuneJuly01 corpus indicated a rather large variance (see Table 5). Moreover, the mean values for the completed and uncompleted dialogs were 0.82 and 0.57 respectively; a t-test indicated that the difference between these means is statistically significant at a very high level of confidence ($p = 7.23 \cdot 10^{-9}$). These results, together with the robust fit, suggest that indeed there is correlation between the CTC-ITC metric and the goodness-of-dialog, and therefore endorse the use of CTC-ITC as a dialog performance metric.

Next, we addressed the slightly more subtle issue of coverage for the models in terms of predictor variable values. The training data for the cost models (the JuneJuly01 corpus) was collected by running the system with a confidence annotator employing the default threshold of 0.5. This implies on average a certain ratio between false-positive and false-negative errors, and it therefore could be argued that the collected data does not allow us to create a model which would reliably extrapolate to the extremes of the operating characteristic, where these predictors would have much different values. An analysis of the distribution of the false-positive and false-negative rates across sessions indicated however that this was not the case. As Figure 8 illustrates, these distributions show that the training set contains a lot of data-points (sessions) which have false-positive and false-negative rates corresponding to the "flat" portion of the operating characteristic ($0 < t < 0.5$).

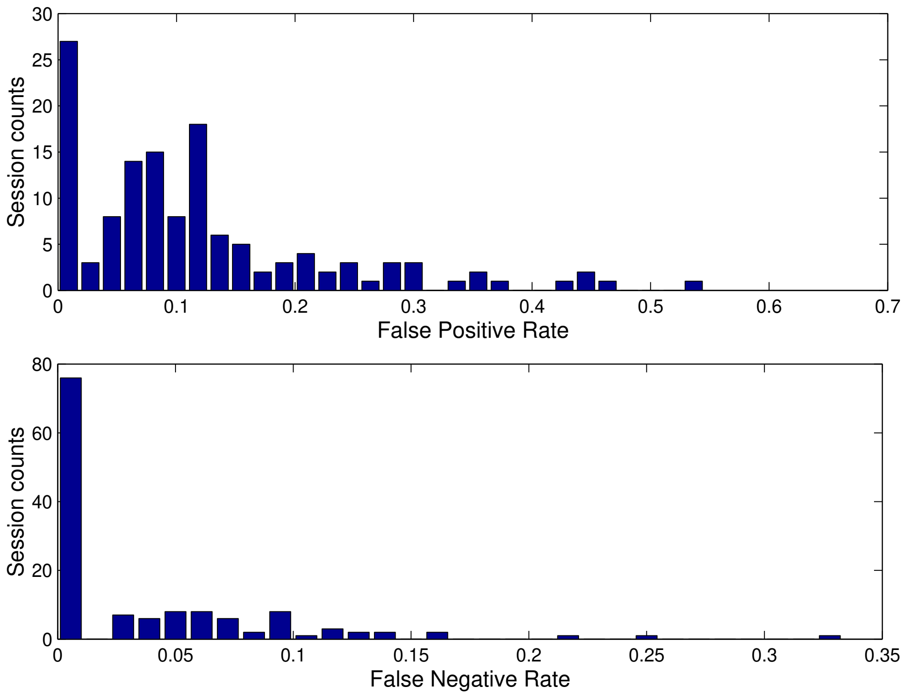


Figure 8: Distribution of false-positive and false-negative rates across sessions

Although the $R^2$ values indicated a strong fit for the models, we performed further robustness checks for the models obtained using dialog efficiency as the target variable. The corpus was randomly divided into two equally sized data-sets, and two separate models were constructed based on these sets. The coefficients obtained are illustrated side-by-side in Table 8. Although some differences are noticeable, they are relatively small and basically lead to the same shape for the cost function in terms of the classification threshold. Plotting the confidence intervals for the cost coefficients as a function of the number of data-points used for regression indicated that more data would possibly further increase the precision of these estimates (the sizes of the 95% cost confidence intervals kept exhibiting a decreasing trend, even after using all the data-points). Nevertheless, changes in cost coefficients up to the size of these intervals are clearly not large enough to account for a significant change in the shape of the cost function.

Experiments were also performed to evaluate the impact of the baseline error rate. A plot of the cost function determined based only on dialogs with a low error rate indicated that in this setting the optimal threshold for the classifier is at zero, equivalent to eliminating the confidence annotator altogether. This observation corroborated our previous results, and seems to indicate that for spoken dialog systems which have a low baseline error rate to

| | **Model 1** | **Model 2** |
|---|---|---|
| $R^2$ | 0.9167 | 0.9683 |
| Constant term | 0.4131 | 0.4137 |
| REC | 0.6300 | 0.6311 |
| FPNC | -0.5914 | -0.4009 |
| FPC | -1.8340 | -2.2860 |
| FN | -1.4871 | -1.2565 |
| TN | -0.6090 | -0.4888 |

Table 8: Comparing cost coefficients for 2 models constructed from a random partition of the data-set

start with, and in which incorrectly captured information can easily be overwritten by the users, a binary utterance-level confidence annotator, although it can lower the classification error rate, is not able to significantly improve dialog efficiency.

The absence of a clear minimum in terms of dialog efficiency as a function of the confidence annotation threshold was indeed an unexpected result. It indicates that, at least for our system, various types of confidence annotation errors trade-off equally in terms of dialog efficiency over a large operating range. We believe that the main explanation for this behavior lies in the overall design of the system. The CMU Communicator is a mixed-initiative system, with a low baseline error rate, and in which incorrectly captured information can be overwritten without a large penalty in terms of efficiency. Moreover, the clarification technique employed by the CMU Communicator is simple and straightforward, and reduces the system's sensitivity to confidence annotation scores. In some sense, our efforts to optimize efficiency in the CMU Communicator via confidence annotation were precluded by an already robust (although not necessarily optimal) system design.

# 5 Conclusion

Speech recognition errors create strong limitations in the development of complex, interactive spoken dialog systems. We believe that robustness can be achieved in these systems by (1) giving them the ability to accurately detect misunderstandings, and (2) developing and employing a set of appropriate recovery techniques when such breakdowns in interaction are detected. In this paper, we have addressed the first problem, that of developing an accurate utterance-level confidence annotator.

The problem was cast as a machine learning classification task. We have shown that, apart from the decoder-level features traditionally used to detect speech recognition errors, a spoken dialog system contains at least two other valuable sources of knowledge: the parsing and the dialog management components. Twelve such features from these 3 different levels were selected and integrated in the confidence annotation process.

Experiments with six different classification techniques have indicated that, although a single best performing classifier cannot be clearly identified, several of them are able to significantly reduce the classification error rate for confidence annotation (50% relative reduction from the majority baseline, and 34% relative reduction from a previously used heuristic rule). We have also verified experimentally that, for continuous-score confidence annotation, density estimation techniques clearly outperform discriminant classification methods.

Subsequently, we have performed what we believe to be the first empirical quantitative assessment of the impact of confidence annotation errors on dialog performance. Several regression models were constructed, relating the counts of various types of errors to different dialog evaluation metrics: efficiency, completion, user-satisfaction. The models fit the data quite well and the relative costs of errors are in accordance to our intuitions. However, when trying to optimally fine-tune the confidence annotator based on these costs, we obtain a surprising result: for a mixed initiative system like the CMU Communicator, false-positive and false-negative errors trade-off equally over a large operating range. A careful analysis of the results and of the factors involved in these experiments led to the conclusion that this effect is most probably due to the already robust (and somewhat insensitive to confidence annotation errors) system design.

Although in our system it does not lead to any significant improvements, we believe that the proposed confidence-error cost estimation approach is an important contribution which remains valid and is applicable in a large number of settings. The results might prove to be radically different (and lead to significant improvements) for other spoken dialog systems. Even for the CMU Communicator, we believe that, once we incorporate more sophisticated confirmation and clarification strategies, the system's sensitivity to confidence annotation will increase, and an optimal confidence annotation threshold could be more clearly identified.

Future research work will concentrate on two main directions: first, more efforts will be directed at further improving the performance of the confidence annotator. The similar, consistent results attained by several classification techniques seem to indicate that we have reached a certain optimal point, given the set of features that we have used. Identifying other useful features turns out to be paramount to further improvements in confidence annotation performance. Another question that we intend to address along the same line is the study of the transferability of these features and of the constructed confidence annotator across different domains.

Once a sufficiently reliable confidence annotation mechanism is constructed, the next research problem is given by the second part of our proposed approach to increased robustness: developing and employing a set of appropriate dialog repair and recovery techniques. Some of the questions under scrutiny are: (1) what is the set of recovery techniques to be used ? (2) what is the best way of integrating and updating beliefs at the dialog management level based on the confidence scores, and (3) how is this information used to optimally trigger the appropriate recovery mechanisms ?

# 6 Acknowledgements

We would like to thank Chun Jin, Paul Carpenter, Daniel Wilson, Rong Zhang for their collaboration on the first part of this work. We would also like to thank Tania Liebowitz and Tina Bennett for their help in the data labelling effort.

This research was sponsored in part by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

## References


[1] Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Shern, R., Lenzo, K., Xu W., Oh, A. “Creating natural dialogs in the Carnegie Mellon Communicator system”, Proceedings of Eurospeech, 1999, 4, 1531-1534.

[2] Rudnicky A., Bennett C., Black A., Chotimongkol A., Lenzo K., Oh A., Singh R. “Task and domain specific modelling in the Carnegie Mellon Communicator system ”, Proceedings of ICSLP 2000 (Beijing, China) Paper G4-0.

[3] Xu W., and Rudnicky A. “Task-based dialog management using an agenda”, ANLP/NAACL 2000 Workshop on Conversational Systems, May 2000, pp. 42-47.

[4] Zue V., et al “JUPITER: A telephone-based conversational interface for weather information”, IEEE Trans. on Speech and Audio Processing, vol.8, no.1, January 2000.

[5] Seneff S., and Polifroni J., ”Dialogue Management in the Mercury Flight Reservation System,” presented at Satellite Dialogue Workshop, ANLP-NAACL, Seattle, April 2000.

[6] Carpenter P., Jin C., Wilson D., Zhang R., Bohus D., Rudnicky A., “Is This Conversation On Track?”, Proceedings of Eurospeech, 2001.

[7] Smith R.W., and Hipp D.R. “Spoken Natural Language Dialog Systems”, Oxford University Press, 1994.

[8] Walker M.A., Fromer J., Fabbrizio G., Mestel C., Hindle D. “Whan can I say ? Evaluating a Spoken Language Interface to Email”, CHI, p. 582-589, 1998.

[9] Gorin A.L., Riccardi G., Wright J. H. “How may I help you?”, Speech Communication, vol.23, no.1/2, pp. 113-127, 1997.

[10] Bertenstam J., Gustafson J., Hunnicutt S., Hgberg J., Lindell R., Neovius L., Serpa-Leitao A., Nord L., and Strm N. “The Waxholm system - a progress report”, Proceedings of Spoken Dialog Systems, Vigso, Denmark, 1995.

[11] Allen J. , Ferguson G., Miller B.W., Ringger E.K. , and Zollo T.S., ”Dialogue systems: From theory to practice in TRAINS-96,” in Robert Dale, Hermann Moisl, and Harold Somers, eds., Handbook of Natural Language Processing, Marcel Dekker, New York, July 2000, pp. 347-376.

[12] Bansal, D., and Ravishankar, M. K. “New Features for Confidence Annotation”, ICSLP-98.

[13] Chase, L. “Error-Responsive Feed back Mechanisms for Speech Recognizers”, Ph.D. Thesis, CMU. 1997.

[14] Cox S., and Rose R. “Confidence Measures for the Switchboard Database”, ICASSP-96.

[15] Kemp T., and Schaaf T. “Estimating Confidence Using Word Lattice”, EuroSpeech-97.

[16] San-Segundo, R., Pellom, B., and Ward, W. “Confidence Measures for Dialogue Management in the CU Communicator System”, ICASSP-2000.

[17] Hazen, T., Burianek, T., Polifroni, J., Seneff, S. "Integrating Recognition Confidence Scoring with Language Understanding and Dialog Modeling", ICASSP-2000.

[18] Walker M., Kamm C., Litman D. "Towards Developing General Models of Usability with PARADISE", Natural Language Engineering, 1998.

[19] Seneff S., Hurley E., Lau R., Pao C., Schmid P., Zue V. "Galaxy-II: A reference architecture for conversational system development", Proceedings ICSLP, Sydney, Australia, 1998.

[20] Huang X., Alleva F., Hon H.-W., Hwang M.-Y., Lee K.-F., and R. Rosenfeld "The SPHINX-II speech recognition system: an overview", Computer Speech and Language, vol.7, no.2, pp 137–148, 1993.

[21] Ward W., and Issar S. "Recent Improvements in the CMU Spoken Language Understanding System", Proceedings of the ARPA Human Language Technology Workshop, 1994, pp. 213-216

[22] Oh A. H., and Rudnicky A. "Stochastic language generation for spoken dialogue systems", ANLP/NAACL 2000 Workshop on Conversational Systems, May 2000, pp. 27-32.

[23] Black A., and Lenzo K. "Limited domain synthesis", Proceedings of ICSLP2000, Beijing, China, 2000.

[24] Chotimongkol A., and Rudnicky. A. "N-best Speech Hypotheses Reordering Using Linear Regression", Proceedings of EuroSpeech 2001, Aalborg, Denmark, September 2001

[25] Heckerman D. "A tutorial on learning with bayesian networks", Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995.

[26] Schapire R.E., and Freund Y. "Experiments with a New Boosting Algorithm", International Conference on Machine Learning, pp. 148-156, 1996

[27] Quinlan J.R., "C4.5: Programs for Machine Learning", Los Altos, California: Morgan Kaufmann

[28] Good I.J., "The Estimation of Probabilities: An Essay on Modern Bayesian Methods", MIT Press, 1965

[29] Jain A.K., Mao J., and Mohiuddin K.M. "Artificial Neural Networks: A Tutorial", IEEE Computer, vol. 29, nr. 3, pp. 31–44, 1996

[30] Burges Christopher J. C. "A Tutorial on Support Vector Machines for Pattern Recognition", Data Mining and Knowledge Discovery, vol.2, no.2, pp.121-167, 1998

[31] Hosmer D.W., and Lemeshow S. "Applied Logistic Regression", Wiley Seried in Probability and Statistics, 2000

[32] Walker M.A., Litman D.J., Kamm C.A., Abella A. "PARADISE: A Framework for Evaluating Spoken Dialogue Agents", Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, 1997.

[33] TellMe web-site, www.tellme.com, as of November 2002

[34] BeVocal web-site, www.heyanita.com, as of November 2002

[35] HeyAnita web-site, www.bevocal.com, as of November 2002