Using Natural Language Processing and Discourse Features to Identify Understanding Errors in a Spoken Dialogue System

Marilyn Walker WALKER@RESEARCH.ATT.COM
Jerry Wright JWRIGHT@RESEARCH.ATT.COM
AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ, 07932-0971, USA

Irene Langkilde ILANGKIL@ISI.EDU
USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA

Abstract

While it has recently become possible to build spoken dialogue systems that interact with users in real-time in a range of domains, systems that support conversational natural language are still subject to a large number of spoken language understanding (SLU) errors. Endowing such systems with the ability to reliably distinguish SLU errors from correctly understood utterances might allow them to correct some errors automatically or to interact with users to repair them, thereby improving the system's overall performance. We report experiments on learning to automatically distinguish SLU errors in 11,787 spoken utterances collected in a field trial of AT&T's *How May I Help You* system interacting with live customer traffic. We apply the automatic classifier RIPPER (Cohen 96) to train an SLU classifier using features that are automatically obtainable in real-time. The classifer achieves 86% accuracy on this task, an improvement of 23% over the majority class baseline. We show that the most important features are those that the natural language understanding module can compute, suggesting that integrating the trained classifier into the NLU module of the *How May I Help You* system should be straightforward.

1. Introduction

Spoken dialogue systems promise efficient and natural access to a large variety of information sources and services from any phone. Several hundred systems that support system-initiative dialogues are currently being field-tested and deployed. However systems that support mixed-initiative, conversational, natural language interaction are still subject to a large number of spoken language understanding (SLU) errors, which have a large impact on the system's performance (Walker et al., 2000b; Walker, 2000). If it were possible to automatically recognize these errors, we could improve the system manually, or even endow the system with the ability to correct some errors automatically or to interact with users to repair them.

Table 1. Sample Successful Dialogue

S1: AT&T How may I help you?
U1: I need to [uh] put a call on my calling card please
S2: May I have your card number, please?
U2: 7 6 5 4 3 2 1 0 9 8 7 6 5 4
S3: What number would you like to call?
U3: 8 1 4 7 7 7 6 6 6 6 (*misunderstood*)
S4: May I have that number again?
U4: 8 1 4 7 7 7 6 6 6 6
S5: Thank you.

The *How May I Help You* (HMIHY) conversational dialogue system developed at AT&T is designed to automatically handle a number of customer care tasks, such as person-to-person dialing, or receiving credit for a misdialed number. As an example of the benefit of SLU error detection, contrast the dialogues in Tables 1 and 2 between human callers and the HMIHY system.[1] HMIHY completed the dialogue in Table 1 successfully. Note that after detecting the SLU error in utterance U3, HMIHY reprompts the user to repeat the calling card number in utterance S4. However, the system frequently **does not** detect that an SLU error has occurred. In Table 2, the system misunderstands the caller's utterance in U2 as a request to make a third-number call. The system continues the dialogue with utterance S3 by asking which number the caller would like to bill the call to. The dialogue ends without completing the user's task.

[1] The phone numbers, card numbers, and pin numbers in the sample dialogues are artificial.

Table 2. HMIHY fails to complete the user's task.

S1: AT&T How may I help you?
U1: (*silence*)
S2: Sorry. Please briefly tell me how I may help you?
U2: I wanta find out how much my bill's worth.
S3: What phone number would you like the call billed to?
U3: *Um* the access number.
S4: Sorry. Could I have that phone number again?
U4: 1 8 hundred 2 2 5 5 2 8 8.
S5: What number would you like to call?
U5: 800 225 5288 (*touchtone input*)
S6: Thank you.

The goal of the work described in this paper is to improve HMIHY's ability to detect such SLU errors. We distinguish three classes of SLU outcomes: RCORRECT, a correctly understood utterance; RPARTIAL-MATCH a partially understood utterance; and RMISMATCH a misunderstood utterance. We describe experiments on learning to automatically distinguish these three classes of SLU outcomes in 11,787 spoken utterances collected in a field trial of AT&T's *How May I Help You* system interacting with live customer traffic. Using ten-fold cross-validation, we show that the resulting SLU error classifier can correctly identify whether an utterance is an SLU error 86% of the time, an improvement of 23% over the majority class baseline. In addition we show that the most important features are those that the natural language understanding (NLU) module can compute, suggesting that it will be straightforward to integrate our SLU error classifier into the NLU module of the HMIHY system.

2. The How May I Help You Corpus

The *How May I Help You* (HMIHY) system has been under development at AT&T Labs for several years (Gorin et al., 1997; Boyce & Gorin, 1996). Its design is based on the notion of *call routing* (Gorin et al., 1997; Chu-Carroll & Carpenter, 1999). In the HMIHY call routing system, services that the user can access are classified into 14 categories, plus a category called *other* for calls that cannot be automated and must be transferred to a human customer care agent (Gorin et al., 1997). Each of the 14 categories describes a different task, such as person-to-person dialing, or receiving credit for a misdialed number. The system determines which task the caller is requesting on the basis of its understanding of the caller's response to the open-ended system greeting *AT&T, How May I Help You?*. Once the task has been determined, the information needed for completing the caller's request is obtained using dialogue submodules that are specific for each task (Abella & Gorin, 1999).

The corpus of dialogues illustrated in Tables 1 and 2 was collected in several experimental trials of HMIHY on live customer traffic (Riccardi & Gorin, 2000; E. Ammicht & Alonso, 1999; Riccardi & Gorin, 2000). These trials were carried out at an AT&T customer care site, with the HMIHY system constantly monitored by a human customer care agent who could take over the call if s/he perceived that the system was having trouble completing the caller's task. The system was also given the ability to automatically transfer the call to a human agent if it believed that it was having trouble completing the caller's task.

The HMIHY system consists of an automatic speech recognizer, a natural language understanding module, a dialogue manager, and a computer telephony platform. During the trial, the behaviors of all the system modules were automatically recorded in a log file, and later the dialogues were transcribed by humans and labelled with one or more of the 15 task categories, representing the task that the caller was asking HMIHY to perform, on a per utterance basis. We will refer to this label as the HUMAN LABEL. The natural language understanding module (NLU) also logged what it believed to be the correct task category. We will call this label the NLU LABEL. This paper focuses on the problem of improving HMIHY's ability to automatically detect when the NLU LABEL is wrong. As mentioned above, this ability would allow HMIHY to make better decisions about when to transfer to a human customer care agent, but it might also support repairing such misunderstandings, either automatically or by interacting with a human caller.

3. Experimental System, Data and Method

The experiments reported here primarily utilize the rule learning program RIPPER (Cohen, 1996) to automatically induce an SLU error classification model from the 11,787 utterances in the HMIHY corpus. Although we had several learners available to us, we had a number of reasons for choosing RIPPER. First, we believe that the if-then rules that are used to express the learned classification model are easy for people to understand and would affect the ease with which we could integrate the learned rules back into our spoken dialogue system. Second, RIPPER supports continuous, symbolic and textual bag features, while other learners do not support textual bag features. Finally, our application utilized RIPPER's facility for changing the loss ratio, since some classes of errors are more catastrophic than others.

However, since the HMIHY corpus is not publicly available, we also ran several other learners on our data in order to provide baseline comparisons on classification accuracy with several feature sets, including a feature set whose performance is statistically indistinguishable from our complete feature set. We report experiments from running

Table 3. Features for Spoken Utterances in the HMIHY Corpus

- **ASR Features**
  - recog, recog-numwords, asr-duration, dtmf-flag, rg-modality, rg-grammar, tempo
- **NLU Features**
  - a confidence measure for all of the possible tasks that the user could be trying to do
  - salience-coverage, inconsistency, context-shift, top-task, nexttop-task, top-confidence, diff-confidence, confpertime, salpertime
- **Dialogue Manager and Discourse History Features**
  - sys-label, utt-id, prompt, reprompt, confirmation, subdial
  - discourse history: num-reprompts, num-confirms, num-subdials, reprompt%, confirmation%, subdialogue%

NAIVE BAYES (Domingos & Pazzani, 1996), MC4 (Kohavi et al., 1997) and an implementation of an INSTANCE-BASED classifier (Duda & Hart, 1973), all available via the MLC++ distribution (Kohavi et al., 1997).

Like other automatic classifiers, RIPPER takes as input the names of a set of *classes* to be learned, the names and ranges of values of a fixed set of *features*, and *training data* specifying the class and feature values for each example in a training set. Its output is a *classification model* for predicting the class of future examples. In RIPPER, the classification model is learned using greedy search guided by an information gain metric, and is expressed as an ordered set of if-then rules.

To apply RIPPER, the utterances in the corpus must be encoded in terms of a set of classes (the output classification) and a set of input features that are used as predictors for the classes. As mentioned above, we distinguish three classes based on comparing the NLU LABEL, the HUMAN LABEL and recognition results for card and telephone numbers: (1) RCORRECT: NLU correctly identified the task and any digit strings were also correctly recognised; (2) RPARTIAL-MATCH: NLU correctly recognized the task but there was an error in recognizing a calling card number or a phone number; (2) RMISMATCH: NLU did not correctly identify the user's task. The RCORRECT class accounts for 7481 (63.47%) of the utterances in the corpus. The RPARTIAL-MATCH accounts for 109 (0.1%) of the utterances, and the RMISMATCH class accounts for 4197 (35.6%) of the utterances.

Next, each utterance is encoded in terms of a set of 43 features that have the potential to be used during runtime to alter the course of the dialogue. These features were either automatically logged by one of the system modules or derived from these features. The system modules that we extracted features from were the acoustic processer/automatic speech recognizer (Riccardi & Gorin, 2000), the natural language understanding module (Gorin et al., 1997), and the dialogue manager, along with a representation of the discourse history (Abella & Gorin, 1999). Because we want to examine the contribution of different modules to the problem of predicting SLU error, we trained a classifier that had access to all the features and compared its performance to classifiers that only had access to ASR features, to NLU features, and to discourse contextual features. Below we describe features obtained from each module. The entire feature set is summarized in Table 3.

The automatic speech recognizer (ASR) takes as input the caller's speech and produces a transcription of what the user said. The ASR features extracted from the corpus were the output of the speech recognizer (*recog*), the number of words in the recognizer output (*recog-numwords*), the duration in seconds of the input to the recognizer (*asr-duration*), a flag for touchtone input (*dtmf-flag*), the input modality expected by the recognizer (*rg-modality*) (one of: none, speech, touchtone, speech+touchtone, touchtone-card, speech+touchtone-card, touchtone-date, speech+touchtone-date, or none-final-prompt), and the grammar used by the recognizer (*rg-grammar*) (Riccardi & Gorin, 2000). We also calculate a feature called *tempo* by dividing the value of the *asr-duration* feature by the *recog-numwords* feature.

The motivation for the ASR features is that any one of them may have impacted recognition performance with a concomitant effect on spoken language understanding. For example, previous work has consistently found *asr-duration* to be correlated with incorrect recognition. The name of the grammar (*rg-grammar*) could also be a predictor of SLU errors since it is well known that the larger the grammar is, the more likely an ASR error is. One motivation for the *tempo* feature is that previous work suggests that users tend to slow down their speech when the system has misunderstood them (Levow, 1998; Shriberg et al., 1992); this strategy actually leads to more errors since the speech recognizer is not trained on this type of speech. The *tempo* feature may also indicate hesitations, pauses, or interruptions, which could also lead to ASR errors. On the other hand, touchtone input in combination with speech, as encoded by the feature *dtmf-flag*, might increase the likelihood of understanding the speech: since the touchtone input is unambiguous it can constrain spoken language understanding.

The goal of the natural language understanding (NLU) module is to identify which of the 15 possible tasks the user is attempting, and extract from the utterance any items of information that are relevant to completing that task, e.g. a

phone number is needed for the task *dial for me*.

Fifteen of the features from the NLU module represent the distribution for each of the 15 possible tasks of the NLU module's confidence in its belief that the user is attempting that task (Gorin et al., 1997). We also include a feature to represent which task has the highest confidence score (*top-task*), and which task has the second highest confidence score (*nexttop-task*), as well as the value of the highest confidence score (*top-confidence*), and the difference in values between the top and next-to-top confidence scores (*diff-confidence*).

Other features represent other aspects of the NLU processing of the utterance. The *inconsistency* feature is an intra-utterance measure of semantic diversity, according to a task model of the domain. Some task classes occur together quite naturally within a single statement or request, e.g. the *dial for me* task is compatible with the *collect call* task, but is not compatible with the *billing credit* task. The *salience-coverage* feature measures the proportion of the utterance which is covered by the salient grammar fragments. This may include the whole of a phone or card number if it occurs within a fragment. The *context-shift* feature is an inter-utterance measure of the extent of a shift of context away from the current task focus, caused by the appearance of salient phrases that are incompatible with it, according to a task model of the domain.

In addition, similar to the way we calculated the *tempo* feature, we normalize the *salience-coverage* and *top-confidence* features by dividing them by *asr-duration* to produce the *salpertime* and *confpertime* features.

The motivation for these NLU features is to make use of information that the NLU module has as a result of processing the output of ASR and the current discourse context. For example, for utterances that follow the first utterance, the NLU module knows what task it believes the caller is trying to complete. The *context-shift* feature incorporates this knowledge of the discourse history, with the motivation that if it appears that the caller has changed her mind, then the NLU module may have misunderstood an utterance.

The function of the dialogue manager is to take as input the output of the NLU module, decide what task the user is trying to accomplish, decide what the system will say next, and update the discourse history. The dialogue manager decides whether it believes there is a single unambiguous task that the user is trying to accomplish, and how to resolve any ambiguity.

Features based on information that the dialogue manager logged about its decisions or features representing the ongoing history of the dialogue might also be useful predictors of SLU errors. Some of the potentially interesting dialogue manager events arise due to low NLU confidence levels which lead the dialogue manager to *reprompt* the user or *confirm* its understanding. A reprompt might be a variant of the same question that was asked before, or it could include asking the user to choose between two tasks that have been assigned similar confidences by the NLU module. For example, in the dialogue in Table 4 the system utterance in S3 counts as a reprompt because it is a variant of the question in utterance S2.

Table 4. HMIHY Dialogue with a more specific reprompt

S1: AT&T How may I help you?
U1: I need credit please.
S2: Okay. What is the reason for the credit?
U2: Miss, uh, Miss, different area code than I needed.
S3: Sorry. I need to know whether you need credit for a wrong number, bad connection or a call that was cut off. (*reprompt*)
U3: It is a wrong number.
S4: What was the number that you dialed?

The features that we extract from the dialogue manager are the task-type label, *sys-label*, whose set of values include a value to indicate when the system had insufficient information to decide on a specific task-type, the utterance id within the dialogue (*utt-id*), the name of the prompt played to the user (*prompt*), and whether the type of prompt was a reprompt (*reprompt*), a confirmation (*confirm*), or a subdialogue prompt (a superset of the reprompts and confirmation prompts (*subdial*)). The *sys-label* feature is intended to capture the fact that some tasks may be harder than others. The *utt-id* feature is motivated by the idea that the length of the dialogue may be important, possibly in combination with other features like *sys-label*. The different prompt features again are motivated by results indicating that reprompts are frustrating for users (Walker et al., 2000b).

The discourse history features included running tallies for the number of reprompts (*num-reprompts*), number of confirmation prompts (*num-confirms*), and number of subdialogue prompts (*num-subdials*), that had been played before the utterance currently being processed, as well as running percentages (*percent-reprompts, percent-confirms, percent-subdials*). The use of running tallies and percentages is based on previous work showing that normalized features are more likely to produce generalized predictors (Litman et al., 1999).

The output of each RIPPER experiment is a classification model learned from the training data. We evaluate the model in several ways. First, we train multiple models using different feature sets extracted from different system modules in order to determine which feature sets are having the largest impact on performance. Second, for each

feature set, the error rates of the learned classification models are estimated using ten-fold cross-validation (Weiss & Kulikowski, 1991), by training on a random 10,608 utterances and testing on a random 1,179 utterances 10 successive times. Third, we report precision, recall and the confusion matrix for classifier trained on all the features tested on a random held-out 20% test set. Fourth, for the classifer trained on all the features, we examine the extent to which we can minimize the error on the error classes RMISMATCH and RPARTIAL-MATCH by manipulating RIPPER's loss ratio parameter. Finally, we compare the results of training other learners on the same dataset with several of the feature sets.

4. Results

Table 5 summarizes our overall accuracy results. The first line of Table 5 represents the accuracy from always guessing the majority class (RCORRECT); this is the BASELINE against which the other results should be compared. The first row, labelled ALL, shows the accuracy based on using all the features available from the system modules (as summarized in Table 3). This classifier can identify SLU errors 23% better than the baseline. The second row of of the table, NLU ONLY, shows that the classifier based only on the NLU features performs statistically as well as the classifier based on all the features. The third row of the table, ASR + DISCOURSE shows that combining the ASR features with the DISCOURSE features produces a significant increase in accuracy over the use of ASR features alone, which however still performs worse than the NLU features on their own. The last two rows, ASR ONLY and DISCOURSE ONLY, indicate that it is possible to do significantly better than the baseline using only the features from the recognizer or from the dialogue manager and the discourse history, but these features on their own cannot do as well at predicting NLU accuracy as the NLU module's own features based on its own calculations.

Table 5. Results for detecting SLU Errors using RIPPER (SE = Standard Error)

Features Used	Accuracy	(SE)
BASELINE (majority class)	63.47 %	
ALL	86.16 %	(0.38)
NLU ONLY	84.80 %	(0.38)
ASR + DISCOURSE	80.97 %	(0.26)
ASR ONLY	78.89 %	(0.27)
DISCOURSE ONLY	71.97 %	(0.40)

Table 6 shows some top performing rules that RIPPER learns when given all the features, which directly reflect the usefulness of the NLU features. Note that some of the rules use ASR features in combination with NLU features such as *salience-coverage*. Table 7 shows some top performing rules when RIPPER has access to only the NLU features, which shows that the features that normalize the NLU features by *asr-duration* are highly useful, as well as task-specific combinations of confidence values and the features that the NLU derives from processing the utterance.

We had also hypothesized that features from the dialogue manager and the discourse history might be useful predictors of SLU errors. Table 5 shows that the discourse history and dialogue manager features can improve over the baseline, but that they do not add significantly to the performance of the NLU ONLY feature set. Table 8 shows some of the best performing rules learned by RIPPER when given only these contextual features. These rules make use of many features that one might expect to indicate users who were experiencing SLU errors, such as the length of the dialogue and the percentage of subdialogues so far in the conversation. However, note that none of the rules for the best performing classifier shown in Table 6 make use of any of these contextual features.

Table 9. Precision and Recall for Random 20% test set using All features

Class	Recall	Precision
Rcorrect	92.22 %	86.92 %
Rmismatch	75.56 %	83.17 %
Rpartial-match	25.0 %	80.00 %

Table 10. Confusion Matrix for Random 20% test set using All features: Rcor = Rcorrect, Rmis = Rmismatch, Rpar = Rpartial-match

KEY	Rcor	Rmis	Rpar
Rcorrect	1090	92	0
Rmismatch	162	504	1
Rpartial-match	2	10	1

We also wished to calculate precision and recall for each category, so we constructed a held-out test set by randomly partitioning the corpus into a fixed 80% of the data for training and the remaining 20% for test. After training on 9,922 utterances using all the features available, we then calculated precision and recall and produced a confusion matrix for the 1,865 utterances in the test set. The results are shown in Tables 9 and 10. Table 9 shows that the classification accuracy rate is a result of a high rate of correct classification for the RCORRECT class, at the cost of a lower rate for RMISMATCH and RPARTIAL-MATCH. However, for the HMIHY application, the classification er-

Table 6. A subset of rules learned by RIPPER when given all the features

if (recog contains "number") $\wedge$ (recog-grammar = Billmethod-gram) $\wedge$ (top-confidence $\leq$ 0.631) **then** *Rpartial-match*.
if (sys-label = DIAL-FOR-ME) $\wedge$ (dtmf-flag = 0) $\wedge$ (asr-duration $\geq$ 5.16) $\wedge$ (recog-grammar = Billmethod-gram) $\wedge$ (asr-duration $\leq$ 5.76) $\wedge$ (recog-numwords $\leq$ 3) **then** *Rpartial-match*.
if (salience-coverage $\leq$ 0.69) $\wedge$ (top-confidence $\leq$ 0.82) $\wedge$ (salpertime $\leq$ 0.009) **then** *Rmismatch*.
if (salience-coverage $\leq$ 0.91) $\wedge$ (confpertime $\leq$ 0.10) $\wedge$ (confpertime $\leq$ 0.075) $\wedge$ (diff-confidence $\leq$ 0.12) $\wedge$ (asr-duration $\leq$ 13.28) **then** *Rmismatch*.
if (top-confidence $\leq$ 0.91) $\wedge$ (confpertime $\leq$ 0.077) $\wedge$ (top-confidence $\leq$ 0.5) $\wedge$ (recog-numwords $\leq$ 10) **then** *Rmismatch*.

Table 7. A subset of rules learned by RIPPER when given only the NLU features

if (top-confidence $\leq$ 0.79) $\wedge$ (vtop-calltype=dial-for-me) $\wedge$ (top-confidence $\leq$ 0.64) $\wedge$ (top-confidence $\geq$ 0.63) then *Rpartial-match*
if (salience-coverage $\leq$ 0.62) $\wedge$ (top-confidence $\leq$ 0.82) **then** *Rmismatch*
if (salience-coverage $\leq$ 0.62) $\wedge$ (confpertime $\leq$ 0.079) **then** *Rmismatch*
if (dial-for-me $\leq$ 0.91) $\wedge$ (confpertime $\leq$ 0.11) $\wedge$ (diff-confidence $\leq$ 0.018) $\wedge$ (dial-for-me $\leq$ 0.529) **then** *Rmismatch*
if (top-confidence $\leq$ 0.91) $\wedge$ (salience-coverage $\leq$ 0.826) $\wedge$ (confpertime $\leq$ 0.10) **then** *Rmismatch*
if (dial-for-me $\leq$ 0.91) $\wedge$ (context-shift $\geq$ 0.309) **then** *Rmismatch*
if (dial-for-me $\leq$ 0.99) $\wedge$ (salpertime $\leq$ 0.105) $\wedge$ (salpertime $\leq$ 0.071) $\wedge$ (billing-credit $\geq$ 0.94) $\wedge$ (top-confidence $\leq$ 0.94) **then** *Rmismatch*

rors that are most detrimental to the system's performance are false acceptances, i.e. cases where the system has misunderstood the utterance but goes ahead with the dialogue without realizing it. This situation was illustrated in Table 2, and it resulted in a task failure. Errors where the system has correctly understood but *reprompts* the user or *confirms* its understanding are much less detrimental to overall performance. Thus we carried out a second set of experiments in which we parameterized RIPPER to avoid this kind of error. The results for the same training and test set are shown in Tables 11 and 12. This parameterization increases the recall of the RMISMATCH class to 82.31%, at the cost of a decrease in recall for the RCORRECT class.

Table 11. Precision and Recall using the ALL feature set, Minimizing False Acceptance

Class	Recall	Precision
Rcorrect	86.72 %	90.55 %
Rmismatch	82.31 %	77.1 %
Rpartial-match	56.25 %	42.86 %

In a final set of experiments, we examine the performance of other learners on our problem since the HMIHY corpus is not publicly available. We ran MC4, an implementation of C4.5 using the MLC++ library (Quinlan, 1993; Kohavi et al., 1997), a NAIVE BAYES learner based on Domingos and Pazzani (1996), and an INSTANCE BASED learner based on Aha (1992); Duda and Hart (1973). A summary of the results is given in Table 13. We include the RIPPER results for these feature sets in Table 13 for ease of comparison.

Table 12. Confusion Matrix for Random 20% test set, Minimizing False Acceptance: Rcor = Rcorrect, Rmis = Rmismatch, Rpar = Rpartial-match

KEY	Rcor	Rmis	Rpar
Rcorrect	1025	156	0
Rmismatch	107	549	11
Rpartial	0	7	9

It was not possible to run these additional learners on the full feature set since they didn't support textual bag features. As comparison points, we ran them on the NLU ONLY feature set, which is our best performing nontextual feature set, and on the DISCOURSE ONLY feature set, which provided a smaller improvement over the baseline. As Table 13 shows, NAIVE BAYES (NBAYES) performs worse than the other learners with either feature set, while the performance of IB is worse than RIPPER with the NLU ONLY feature set, and better than RIPPER with the DISCOURSE ONLY feature set. The MC4 learner performs better than RIPPER

Table 8. A subset of rules learned by RIPPER when given only the discourse and dialogue features

if (utt-id $\geq$ 2) $\wedge$ (sys-label=DIAL-FOR-ME) $\wedge$ (percent-subdials $\leq$ 0.333) **then** *Rmismatch*
if (utt-id $\geq$ 3) $\wedge$ (sys-label=DIAL-FOR-ME) $\wedge$ (reprompt=reprompt) **then** *Rmismatch*
if (utt-id $\geq$ 3) $\wedge$ (subdial=not-subdial) $\wedge$ (prompt=collect-no) $\wedge$ (num-subdials $\geq$ 1) **then** *Rmismatch*
if (utt-id $\geq$ 2) $\wedge$ (sys-label=DIAL-FOR-ME) $\wedge$ (percent-subdials $\leq$ 0.333) **then** *Rmismatch*

Table 13. Classification accuracies for identifying SLU Errors for several learners (SE = Standard Error)

Features Used	Accuracy	(SE)
BASELINE (majority class)	63.47 %	
NBAYES, DISCOURSE ONLY	63.72 %	(0.39) %
RIPPER, DISCOURSE ONLY	71.97 %	(0.40) %
IB, DISCOURSE ONLY	75.30 %	(0.33) %
MC4, DISCOURSE ONLY	76.20 %	(0.35) %
NBAYES, NLU ONLY	77.30 %	(0.34) %
IB, NLU ONLY	82.65 %	(0.28) %
RIPPER, NLU ONLY	84.80 %	(0.38) %
MC4, NLU ONLY	85.21 %	(0.42) %

on the DISCOURSE ONLY features and comparable to RIPPER on the NLU ONLY features.

5. Discussion and Future Work

The results of training various classifiers to detect SLU errors show that: (1) All feature sets significantly improve over the baseline; (2) Using features from all the system modules, we can improve the identification of SLU errors by 23% over the majority class baseline; (3) The NLU features alone can perform as well as all the features in combination; (4) RIPPER performs much better than NAIVEBAYES, comparably to IB, and slightly worse than MC4, depending on the feature set.

This research was motivated by earlier work on identifying problems with a spoken dialogue at the dialogue level. Litman et al. (1999) examined the problem of predicting whether the percentage of misunderstood utterances was greater than a threshhold. Obviously, if one could accurately predict whether an utterance had been misunderstood, then predicting whether the percentage was greater than a threshhold would be a trivial problem.[2] Walker et al. (2000a) examined the problem of predicting when a dialogue in the HMIHY corpus is *problematic*, where a problematic dialogue was defined as one where a human customer care agent who was monitoring the dialogue took over the dialogue, or where the user hung up, or the system believed it had completed the user's task but in fact did not. Walker et al. (2000a) showed that the ability to correctly detect whether individual utterances in the dialogue were SLU errors could improve raw accuracy for predicting problematic dialogues by 7%.

Levow (1998) applied similar techniques to learn to distinguish between utterances in which the user originally provided some information to the system, and *corrections*, which provided the same information a second time, following a misunderstanding. Levow utilized features such as duration, tempo, pitch, amplitude, and within-utterance pauses, with the finding that the durational features were the most important predictors.

A similar approach was also used by Hirschberg et al. (1999) to predict recognition errors. Hirschberg et al. (1999) used prosodic features in combination with acoustic confidence scores to predict recognition errors in a corpus of 2067 utterances. Hirschberg etal. report a best-classifier accuracy of 89%, which is a 14% improvement over their baseline of 74%. Even without acoustic confidence scores, our best-classifier accuracy is 86%, a 23% improvement over our baseline of 63%.

Previous work on the HMIHY task has utilized a different corpus of HMIHY utterances than those used here to examine the contribution of acoustic confidence models from ASR to spoken language understanding (Rose & Riccardi, 1999; Rose et al., 1998). Rose, Riccardi and Wright (1998) showed that understanding accuracy could be increased by 23% with the availability of ASR confidence scores. While these results are not directly comparable to those here, both because of the different corpus and because they focused on a different problem, these results suggest that ASR con-

[2] Note that Litman et al. (1999) discuss their work in terms of predicting misrecognition performance rather than misunderstanding performance. This may be partly due to the fact that much of their corpus consisted of user utterances containing only a single word, or only a few words. In this situation, recognition is identical to understanding. In the case of HMIHY where the callers are initially often not aware they are talking to a system, understanding is considerably more complex. Although recognition errors tend to be correlated with understanding errors, experiments with HMIHY show that recognition word accuracy may go down, while understanding concept accuracy increases (Rose et al., 1998).

fidence scores could also be useful for predicting SLU errors. However when the HMIHY corpus was collected, ASR confidence scores were not available.

In future work, we intend to examine the contribution of ASR confidence scores to identifying SLU errors, to investigate whether we can use the SLU classifier results to re-rank ASR outputs, and to integrate the learned rulesets into the NLU module of the HMIHY dialogue system in the hope of demonstrating an improvement in the system's overall performance.

References

Abella, A., & Gorin, A. (1999). Construct algebra: An analytical method for dialog management. *Proceedings of Thirty Seventh Annual Meeting of the Association for Computational Linguistics*.

Aha, D. W. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies, 36*, 267–287.

Boyce, S., & Gorin, A. L. (1996). User interface issues for natural spoken dialogue systems. *Proceedings of International Symposium on Spoken Dialogue* (pp. 65–68).

Chu-Carroll, J., & Carpenter, B. (1999). Vector-based natural language call routing. *Computational Linguistics, 25-3*, 361–387.

Cohen, W. (1996). Learning trees and rules with set-valued features. *Fourteenth Conference of the American Association of Artificial Intelligence*.

Domingos, P., & Pazzani, M. (1996). Beyond independence: Conditions for the optimality of the simple bayesian classifier. *Proceedings of the 13th International Conference on Machine Learning*.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. John Wiley & Sons.

E. Ammicht, A. G., & Alonso, T. (1999). Knowledge collection for natural language spoken dialog systems. *Proceedings of the European Conference on Speech Communication and Technology*.

Gorin, A., Riccardi, G., & Wright, J. (1997). How may I Help You? *Speech Communication, 23*, 113–127.

Hirschberg, J. B., Litman, D. J., & Swerts, M. (1999). Prosodic cues to recognition errors. *Proceedings of the Automatic Speech Recognition and Understanding Workshop*.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1997). Datamining using MLC++, a machine learning library in C++. *International Journal of AI Tools, 6-4*, 537–566.

Levow, G.-A. (1998). Characterizing and recognizing spoken corrections in human-computer dialogue. *Proceedings of the Thirty Sixth Annual Meeting of the Association of Computational Linguistics* (pp. 736–742).

Litman, D. J., Walker, M. A., & Kearns, M. J. (1999). Automatic detection of poor speech recognition at the dialogue level. *Proceedings of the Thirty Seventh Annual Meeting of the Association of Computational Linguistics* (pp. 309–316).

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Riccardi, G., & Gorin, A. (2000). Spoken language adaptation over time and state in a natural spoken dialog system. *IEEE Transactions on Speech and Audio Processing, 8*, 3–10.

Rose, R. C., & Riccardi, G. (1999). Automatic speech recognition using acoustic confidence conditioned language models. *Proceedings of the European Conference on Speech Communication and Technology*.

Rose, R. C., Yao, H., Riccardi, G., & Wright, J. (1998). Integration of utterance verification with statistical language modeling and spoken language understanding. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*.

Shriberg, E., Wade, E., & Price, P. (1992). Human-machine problem solving using spoken language systems (SLS): Factors affecting performance and user satisfaction. *Proceedings of the DARPA Speech and NL Workshop* (pp. 49–54).

Walker, M., Langkilde, I., Wright, J., Gorin, A., & Litman, D. (2000a). Learning to predict problematic situations in a spoken dialogue system: Experiments with How May I Help You? *Proceedings of the North American Meeting of the Association for Computational Linguistics*.

Walker, M. A. (2000). An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*.

Walker, M. A., Kamm, C. A., & Litman, D. J. (2000b). Towards developing general models of usability with PARADISE. *Natural Language Engineering: Special Issue on Best Practice in Spoken Dialogue Systems*.

Weiss, S. M., & Kulikowski, C. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and exper t systems*. San Mateo, CA: Morgan Kaufmann.