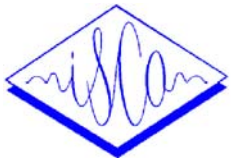

6th SIGdial Workshop on
Discourse and Dialogue
Lisbon, Portugal
September 2-3, 2005

Sorry, I Didn't Catch That! – An Investigation of Non-understanding Errors and Recovery Strategies

Dan Bohus
Carnegie Mellon University
Pittsburgh, PA, 15213
dbohus@cs.cmu.edu

Alexander I. Rudnicky
Carnegie Mellon University
Pittsburgh, PA, 15213
air@cs.cmu.edu

Abstract

We present results from an extensive empirical analysis of non-understanding errors and ten non-understanding recovery strategies, based on a corpus of dialogs collected with a spoken dialog system that handles conference room reservations. More specifically, the issues we investigate are: what are the main sources of non-understanding errors? What is the impact of these errors on global performance? How do various strategies for recovery from non-understandings compare to each other? What are the relationships between these strategies and subsequent user response types, and which response types are more likely to lead to successful recovery? Can dialog performance be improved by using a smarter policy for engaging the non-understanding recovery strategies? If so, can we learn such a policy from data? Whenever available, we compare and contrast our results with other studies in the literature. Finally, we summarize the lessons learned and present our plans for future work inspired by this analysis.

1 Introduction

One of the most important challenges facing spoken language interfaces today is their brittleness when faced with understanding errors. The problem is present across all domains and interaction types, and arises primarily from the inherent unreliability of the speech recognition process. The recognition difficulties are further exacerbated by the conditions under which these systems typically operate: spontaneous speech, large vocabularies and user populations, and large variability in input line quality. In these settings, average word-error-rates of 20-30% (and up to 50% for non-native speakers) are quite common. Unless mediated by better error awareness and robust recovery mechanisms, these errors exert a strong negative influence on the overall performance of spoken dialog systems (Sanders et al, 2002; Walker et al, 2002), and severely limit the naturalness of the interaction and the complexity of the tasks that can be addressed.

Left unchecked, speech recognition errors can lead to two types of *understanding errors* in a spoken dialog system: *misunderstandings* and *non-understandings*. In a *misunderstanding*, the system obtains an incorrect interpretation of the user's turn. In contrast, in a *non-understanding*, the system fails to obtain any interpretation of the input.

In this paper, we focus our attention on non-understandings. If for misunderstandings detection is a key problem (San-Segundo et al, 2000; Litman et al, 2000; Carpenter et al, 2001), and the set of recovery strategies is limited and fairly well understood (e.g. explicit and implicit confirmation (Krahmer et al, 1999)), for non-understandings the situation is almost the opposite. By definition, systems know when a non-understanding has happened. However, a mechanism for diagnosing the source of the non-understanding is largely missing. Moreover, the number of potential recovery strategies is significantly larger (see Table 1) and the relative trade-offs between them are less well understood. This further increases the difficulty of selecting the right recovery strategy at runtime. Most systems use a limited number of non-understanding recovery strategies in conjunction with uninformed, simple heuristic rules for engaging them. For instance, a system might apologize and repeat its question on the first non-understanding, provide more help on the second non-understanding, and transfer the user to a human operator if a third consecutive non-understanding occurred.

As a first step towards better error handling for non-understandings, we have conducted an empirical study of these errors and of ten recovery strategies based on

data collected in a mixed-initiative, task-oriented spoken dialog system. More specifically, the questions we have investigated are:

- *What are the main sources of non-understanding errors (and what are their relative frequencies)?*
- *How large is the impact of non-understandings on global dialog performance?*
- *How do various strategies for recovering from non-understandings compare to each other?*
- *What are the relationships between each strategy and subsequent user behaviors, and which behaviors are more likely to lead to successful recovery?*
- *Can global dialog performance be improved by using a smarter policy for engaging the non-understanding recovery strategies?*
- *If yes, can we learn a better policy from data?*

We begin by describing the data collection experiment which provided the corpus of dialogs used in this investigation. Then, over the following six sections, we address in turn each of the questions raised above. Whenever possible, we compare our findings to other results previously reported in the literature, in an effort to shed more light on the generalizability of these results across different domains. Finally, in Section 9 we summarize the lessons we learned from this investigation and the ideas it inspired for future work.

2 Experiment and Corpus

2.1 Data Collection Experiment

System. The data was collected through a user study in which 46 participants, mostly undergraduate and staff personnel on campus interacted with RoomLine (RoomLine, 2003), a spoken dialog system for making conference room reservations. RoomLine is a phone-based mixed-initiative system which has access to live information about the schedules and characteristics (e.g. size, location, A/V equipment) of 13 conference rooms in two buildings on campus. To make a room reservation, the system finds the list of available rooms that satisfy an initial set of user-specified constraints, and engages in a follow-up negotiation dialog to present this information to the user and identify which room best matches their needs. Sample conversations with the system are available online (RoomLine, 2003).

The system uses two parallel SPHINX-II recognition engines, configured with telephone-based acoustic models and a trigram statistical language model (the dictionary size is 1049). The resulting top hypothesis from each engine is parsed using the Phoenix robust parser (Ward and Isaar, 1994). Subsequently, semantic confidence scores are computed for each hypothesis. The winning hypothesis is forwarded to the RavenClaw-based dialog manager (Bohus and Rudnicky, 2003). For output, the system uses a template-based language generation module and the Theta synthesizer (Theta, 2004).

The system was equipped with ten different *strategies* for recovering from non-understandings, described and illustrated in Table 1. By *strategy* we denote a simple, single-turn action that the system can take to attempt recovery. A number of these strategies, such as asking the user to repeat or rephrase, reissuing the system prompt or providing various levels of help are often encountered in spoken dialog systems. Two strategies that we would like to draw the reader's attention upon are *Yield* and *MoveOn*. In the *Yield* strategy, the system remains silent, as if it did not hear the user's response, and hence implicitly signals a communication problem. In the *MoveOn* strategy, the system ignores the problem altogether and tries to advance the task by moving on to a different question. Note that this is possible only at certain points in the dialog, where an alternative dialog plan for achieving the same goals is available. For instance, in the case illustrated in Table 1, the *MoveOn* strategy gives up on trying to find whether the user wants a small or a large room, and starts suggesting rooms one by one. In other cases, the system would try to advance the dialog by using a simpler question, for instance asking "For which day do you need the room?" instead of "How can I help you?"

Experimental design. The user study was designed as a between-groups experiment, with two conditions: *control* and *wizard*.

Participants in the *control* condition interacted with a version of the RoomLine system which used an uninformed (random) policy to engage the non-understanding recovery strategies: each time a non-understanding happened, the system randomly chose one of the ten available strategies.

Participants in the *wizard* condition interacted with a modified Wizard-of-Oz version of the same system. In this version, each time a non-understanding happened a human wizard decided which one of the ten recovery strategies should be used. In all other aspects, this system was identical with the system used in the control condition. The wizard had live access to the user's speech. Several other system state variables were presented to the wizard via a graphical user interfaces (e.g. recognition result, confidence score, semantic parse). When a non-understanding occurred, the wizard selected which strategy should be used through the GUI, and the decision was communicated back to the system. The wizard had to make this decision during a relatively short time interval (1-2 seconds) in order to maintain the illusion that the users were interacting with an autonomous system. A single wizard, the first author of this paper, was employed throughout the whole experiment. The wizard had very good knowledge of the system's functionality and of the domain.

	S: Would you like a small room or a large one? U: [Non-Understanding]
1	**AskRepeat (AREP)** [system asks the user to repeat the non-understood utterance] S: Can you please repeat that?
2	**AskRephrase (ARPH)** [system asks the user to rephrase the non-understood utterance] S: Can you please try to rephrase that?
3	**Reprompt (RP)** [system repeats the previous prompt] S: Would you like a small room or a large room?
4	**DetailedReprompt (DRP)** [system repeats a more verbose version of the previous prompt] S: I'm sorry, I don't think I understood you correctly. Right now I'm trying to find out if you would prefer a small room or a large one.
5	**Notify (NTFY)** [system merely notifies the user that a non-understanding has occurred] S: Sorry, I'm not sure I understood what you said …
6	**Yield (YLD)** [system remains silent, and thus implicitly notifies the user that a non-understanding has occurred] S: Ø
7	**MoveOn (MOVE)** [system advances the task by moving on to a different question] S: Sorry, I didn't catch that. One choice would be Wean Hall 7220. This room can accommodate 20 people and has a whiteboard and a projector. Would you like a reservation for this room?
8	**YouCanSay (YCS)** [system tell the user what he or she can say at this point in the dialog] S: Sorry, I didn't catch that. Right now I'm trying to find out if you would prefer a small room or a large one. You can say 'I want a small room' or 'I want a large room'. If the size of the room doesn't matter to you, just say 'I don't care'.
9	**TerseYouCanSay (TYCS)** [a terser version of the YouCanSay strategy] S: Sorry, I didn't catch that. You can say 'I want a small room' or 'I want a large room'. If the size of the room doesn't matter to you, just say 'I don't care'.
10	**FullHelp (HELP)** [system provides a longer help message which includes an explanation of the current state of the dialog, as well as what the user can say at this point] S: I'm sorry, I don't think I understood you correctly. So far I have found five conference rooms available matching your constraints. Right now I'm trying to find out if you would prefer a small room or a large room. You can say 'I want a small room' or 'I want a large room'. If the size of the room doesn't matter to you, just say 'I don't care'.

Table 1. Ten strategies for recovering from non-understandings

The experimental design described above satisfies two needs. On one hand, we wanted to be able to comparatively evaluate the ten recovery strategies, when engaged in an uninformed fashion. This analysis can be performed based on data collected in the *control* condition, where the system randomly chooses which strategy to use. The results are discussed in detail in Sections 5 and 6. At the same time, we wanted to verify whether or not a better policy for engaging the ten strategies (implemented in this case by the human wizard) can significantly improve performance. The results of this comparative analysis are presented in Section 7.

At this point we would like to briefly comment on the decision to give the wizard full access to the live user speech. This puts the wizard in an apparently privileged position when compared to a system that would have to make the same recovery decisions (e.g. the system does not accurately know what the user says, especially during non-understandings). However, recall that our goal is only to show that a better recovery policy exists, and *not* to prove that this particular policy can be learned or implemented by the system. Without access to the user's speech, the decision making task might have been too difficult for the wizard, especially given the response-time constraints. In this case, a negative result, i.e. the lack of detectable differences in the performance of the two policies, would not be very informative. On the other hand, a negative result obtained when the wizard has full access to the user's speech would cast more serious doubts about the existence of a better non-understanding recovery policy.

Participants. 46 subjects, mostly undergraduate students and staff personnel on campus, participated in the data collection experiment. The participants had only marginal prior experience with spoken language interfaces (some of them had previously interacted with phone-based customer-service interactive systems). We randomly assigned the participants into two groups corresponding to the *control* and *wizard* conditions. At the same time, a balance was maintained between groups in terms of the participants' gender and whether or not their first language was north-American English.

Tasks and Experimental Procedure. Each participant attempted a maximum of 10 scenario-based interactions with the system, within a set time period of 40 minutes. The same 10 scenarios were presented in the same order to all participants. The scenarios were designed to cover all the important aspects of the system's functionality and had different degrees of difficulty. To avoid language entrainment, the scenarios were presented graphically. Descriptions of the 10 scenarios as well as a concrete example of the graphical representation are available online (Bohus, 2005).

After completing their interactions with the system, the participants filled in a SASSI questionnaire (Hone and Graham, 2000) containing 35 questions grouped in 6 factors: response accuracy, likeability, cognitive demand, annoyance, habitability, and speed. Additionally, participants were asked to describe what they liked most, what they liked least and what would be the first thing they would change in the system.

2.2 Corpus Statistics and Annotations

The corpus of dialogs collected in this experiment (including both the *control* and *wizard* conditions) contains 449 sessions and 8278 user turns. In Table 2 we present a number of additional descriptive statistics. Since pronounced differences exist on a large number of metrics between native and non-native users, we also present the breakdown of the figures in these two populations.

	Total	Native	Non-native
# Subjects	46	34	12
# Sessions	449	338	111
# Turns	8278	5783	2495
Word-error-rate	25.6%	19.6%	39.5%
Concept-error-rate	35.7%	26.3%	57.6%
% Non-understandings	17.0%	13.4%	25.2%
% Misunderstandings	13.5%	9.8%	22.5%
Task success rate	75.1%	85.2%	44.1%

Table 2. Overall corpus statistics

The user speech data was orthographically transcribed by a human annotator, and subsequently checked by a second annotator. The transcriptions include annotations for various human and non-human noises in the audio signal. Based on these transcriptions, a number of additional annotations were created. At the turn level, we manually labeled:

- *Concept transfer* and *misunderstandings*: each user turn was annotated with the number of concepts that were correctly and incorrectly transferred from the user to the system; each turn with at least one incorrectly transferred concept was automatically labeled as a *misunderstanding*;
- *Transcript grammaticality*: each user turn was manually annotated as either *in-grammar*, *out-of-grammar*, *out-of-application-scope* or *out-of-domain* (for a discussion, see Section 3);
- *User responses to non-understandings:* the user response types following non-understandings were labeled using a tagging scheme first introduced by Shin and Narayanan (2002);
- *Corrections*: each turn in which the user was attempting to correct a system understanding error was flagged as a correction, as in (Swerts et al, 2000);

At the session level, we labeled task completion.

3 Sources of Understanding Errors

We now turn our attention to the first question: ***what are the main sources of non-understandings, and what are their relative frequencies?***

While the main focus of this paper is on non-understandings, the analysis we present in this section covers sources of understanding errors in general, i.e. both misunderstandings and non-understandings. To avoid potential biases introduced by the wizard's recovery policy, the analysis was conducted using only data from the *control* condition, where the recovery strategies were engaged in an uninformed fashion.

We anchor our error source analysis in the grounding model inspired by Clark (1996) and used by Paek and Horvitz (2000) in the Conversational Architectures project, illustrated in Figure 1. In this model, participants coordinate on 4 different levels to achieve mutual understanding in conversation. In the context of human-computer interaction, the model also illustrates the flow of information from the user to the system. At the conversation level, the user has a high-level goal, which subsequently acquires a corresponding semantic, lexical and eventually an acoustic representation in the lower levels. The acoustic signal then passes through a noisy channel, and arrives at the system side. Here, a series of chained components (speech recognition, language understanding, and discourse interpretation) are used to progressively reconstruct the user's higher level goal

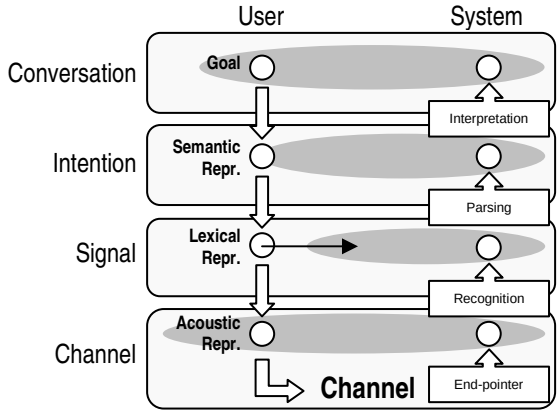

Figure 1. Grounding in communication

from the incoming acoustic signal.

Understanding errors typically occur due to mismatches at different levels between the expressed form of the user's intent and the system's modeling abilities. For example, at the conversation level, the user might not be aware of certain system limitations and might try to formulate a goal which the system cannot handle. In this case it will be impossible for the system to correctly reconstruct the user's goal, and we will have an understanding error. Similarly, at the signal level, mismatches between a user's pronunciation style and the system's acoustic models can lead to speech recognition errors, and ultimately to understanding errors. This view of understanding errors highlights two complementary approaches that can be used to mitigate the mismatches. One is to create models which can provide better coverage, while still maintaining good performance. The other is to steer the user's responses into the space covered by the system's models.

Based on the level at which the mismatch occurs, we identify the following sources of errors:

- ***Out-of-Application*** [Conversation Level]**:** The user's utterance falls outside the application's functionality. These errors can be further divided into *out-of-domain* utterances (e.g. the user asks the room-reservation system about the weather), and *out-of-application-scope* utterances, i.e. utterances which express in-domain goals which the system is however not able to handle (e.g. the user asks if a conference room has windows);
- ***Out-of-Grammar*** [Intention Level]: The user's utterance is within the domain and scope of the application, but outside of the system's semantic grammar (e.g. the user says "erase reservation", which is not in the system's grammar; the system could have handled the request had the user said "cancel reservation" or "delete reservation", which are in the system's grammar);
- ***ASR Error*** [Signal Level]**:** The user's utterance is within the application's domain, scope and grammar, but is not recognized correctly due to acoustic or statistical language modeling mismatches (e.g. the user says "Thursday morning" but this is misrecognized as "Friday morning");
- ***End-pointer Error*** [Channel Level]**:** The end-pointer is not able to correctly segment the incoming audio signal (e.g. it truncates the utterance or sends an empty utterance into the input line)

Figure 2 illustrates the breakdown of non-understandings and misunderstandings by error source. The majority of errors originate at the Signal (i.e. speech recognition) level. At the same time, a large number of non-understandings, and a smaller but still significant number of misunderstandings are caused by *out-of-application* and *out-of-grammar* utterances.

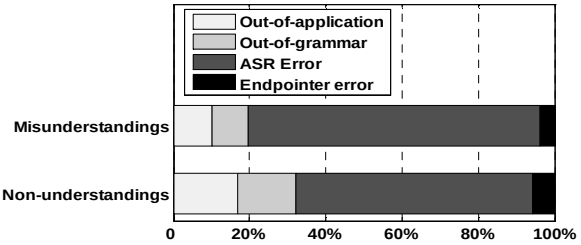

Figure 2. Breakdown of non-understandings and misunderstandings by error source

The *out-of-application* errors encountered in our data consist almost entirely of *out-of-application-scope* utterances. These utterances are in-domain, but they refer to inexistent application functionality (the lack of out-of-domain utterances is most likely due to the scenario-driven nature of the interactions). A closer inspection of these errors revealed that they subsume about an equal number of requests for inexistent task-level functionality (e.g. "I need a room for Monday or Tuesday" – the system does not handle "or" requests), and requests for inexistent meta functionality, such as "go back!" or various types of corrections (e.g. "You got the wrong day!", "Change the date!", "The time is wrong", etc).

Together with the *out-of-grammar* utterances, the *out-of-application* utterances reflect one facet of an existing mismatch between user and system at the intention and conversation levels. A second interesting facet, revealed through an analysis of the transcripts, is that there are certain aspects of system functionality which are *never* (or *very rarely*) addressed by the users. For instance, although the users were told during the briefing that they can say "*Help*" to the system at any time, this function was invoked in only in 7 of 226 sessions. Other types of help commands like "*where are we?*", "*what can you do?*", "*what can I say?*", "*interaction tips*", although available at all times were not discovered by the users and therefore were never used. We found similar examples with respect to task-level functionality, for commands like "*tell me all the rooms*", "*I want a smaller / larger room*", "*I don't care*" (about room size), "*how big is this room*", "*tell me about this room*", etc. This reflects the fact that, apart from out-of-grammar errors, users are also not aware of the full functionality of the application.

The fairly large number of *out-of-application* and *out-of-grammar* utterances suggests that the number of non-understandings can potentially be reduced by better informing the users about the application capabilities and boundaries and steering them into this space. How exactly this shaping can be performed remains an open research issue (Tomko, 2004). We will return to this issue in our discussion from Section 9.

The majority of non-understandings – 62% (and even more so for misunderstandings – 77%) originate at the speech recognition level. Here, a large number of

contributing factors can be identified, but more precise blame assignment is harder to perform. For instance, non-native accents have a significant impact on ASR performance: average WER is 20.7% for natives, versus 42.3% for non-natives. Ambient noises also have a pronounced effect on recognition performance: average WER for noisy utterances is 32.8% > 25.1% for noise-free utterances. Other factors, such as speaking rate, user frustration, hyper-articulation, have been showed to correlate with recognition accuracy (Choularton, 2005).

Rejections. The discussion so far has focused on *genuine non-understandings*, i.e. situations in which the system was not able to extract any meaningful information from the user's turn. However, our dialog manager also uses a rejection mechanism to guard against potential misunderstandings: if the system has obtained an interpretation of the user's input, but the confidence score is below a preset threshold, then the utterance will be rejected by the dialog manager. These rejected utterances will also appear as non-understandings at the dialog management level. Figure 3 illustrates the ratios of non-understandings and misunderstandings, as computed before and after the rejection mechanism. After rejections, the total ratio of non-understandings grows by 7.1% absolute from 10.1% to 17.2%. About 40% of the rejections (2.9% of the total number of turns, and 17% of the total number of non-understandings) are false-rejections, i.e. utterances correctly understood but falsely rejected because of a low confidence score. The relatively high false rejection rate contributes significantly to the total number of non-understandings, on par with other sources of errors. The false-rejection rate can be lowered by building better confidence annotators, or by tuning the rejection threshold to the domain. In (Bohus and Rudnicky, 2005), we describe a data-driven method for optimizing the rejection process in light of domain and dialog-state-specific tradeoffs.

4 Impact of Non-understandings on Dialog Performance

We now turn our attention to the second question: ***what is the impact of non-understanding errors on global dialog performance?*** Again, we only used the data from the *control* condition in our analysis.

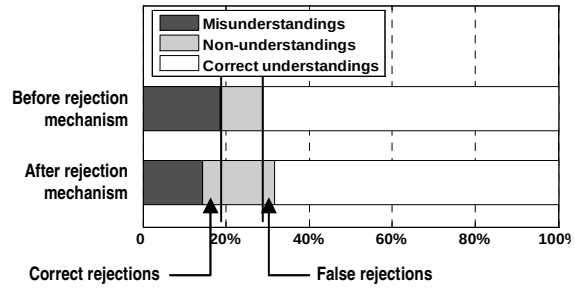

Figure 3. Misunderstandings and non-understandings before and after rejections

To address this question, we constructed a logistic regression model (Myers et al., 2001) which relates the frequency of non-understandings in a dialog to the probability of task success. The same approach can be used for studying the impact on other global performance metrics.

$$P(TS=1)=\frac{1}{1+e^{-(\alpha+\beta\cdot FNON)}}$$

The independent variable is the frequency of non-understandings in a session (***FNON***), and the dependent variable is the binary task success indicator (***TS***). Each data-point corresponds to an entire dialog session.

We fitted a model using 205 dialog sessions. Sessions with less than 3 turns and sessions with differences between perceived and objective task completion were eliminated. The fitted model increased the average data log-likelihood from the majority baseline of -0.5200 to -0.4306 ($p<10^{-4}$ in a likelihood-ratio test), indicating that there is indeed an effect of the frequency of non-understandings on task success. Figure 4 illustrates the expected probability of task success, as predicted by the model. The plot shows that when the frequency of non-understandings is between 0%-10%, the impact on task success is relatively minor. However, as the frequency of non-understandings exceeds 10%, the expected probability of task success starts to drop faster: an increase of the frequency of non-understandings from 10% to 30% reduces the expected chance of success from 90% to 52%.

Apart from non-understandings, misunderstandings represent a second important contributor to breakdowns in interaction. To assess the relative costs of these two types of errors with respect to task success, we extended the model described above to include the frequency of misunderstandings as a second independent variable (***FMIS***). As expected, the new model predicts task success even better: the average log-likelihood of the data was further increased to -0.2795 ($p<10^{-4}$). The estimated regression coefficients, together with their associated standard errors and p-values are illustrated in Table 3. The resulting average cost for misunderstandings (-16.62) is 2.24 times higher than the average cost for non-understandings (-7.41). The result confirms that the rule-of-thumb that "misunderstandings cost twice as much as non-understandings" holds in our domain. While the relative costs of these errors can vary across different domains, and even across different dialog states within the same system, the proposed regression approach can be used to establish these costs in a principled manner (see also Bohus and Rudnicky, 2005).

Finally, we analyzed the impact of *recovery rate* on task success. We say that a strategy has successfully recovered from a non-understanding if the following

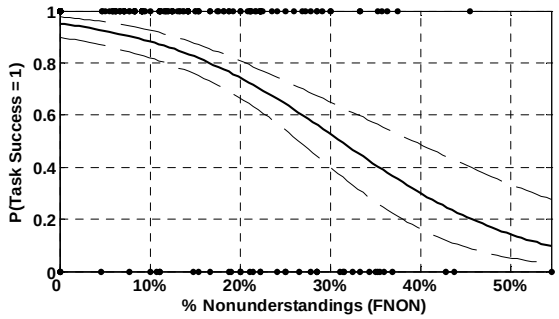

Figure 4. Expected probability of task success (and confidence bounds) at different frequencies of non-understandings

	Coefficients	S.E.	p-value
Const	5.28	0.70	< 0.0001
FNON	-7.41	2.09	0.0004
FMIS	-16.62	2.74	< 0.0001

Table 3. Regression coefficients for a task success model using the frequency of non-understandings and misunderstandings as the independent variables

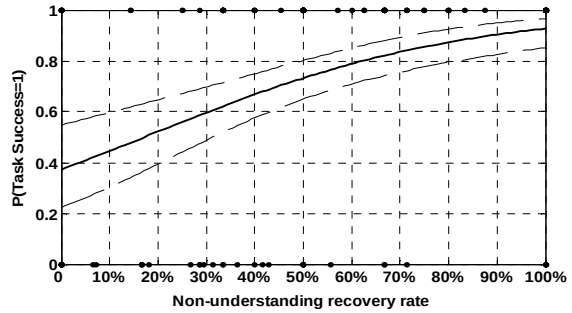

Figure 5. Expected probability of task success (and confidence bounds) at different non-understanding recovery rates

user turn is correctly understood by the system (i.e. it is not a non-understanding and it is not a misunderstanding). The *average non-understanding recovery rate* is then defined as the ratio of successful recoveries, with respect to the total number of attempts to recover. Again, a significant effect on task success was detected ($p<10^{-4}$). The dependence is illustrated in Figure 5. As this figure shows, the impact of the recovery rate on performance is greatest when the recovery rate is below 60-70%, and becomes less significant as we pass that limit.

While it is to be expected that non-understandings and the associated recovery rate have an effect on global performance, the analyses that we have performed quantify this effect and provide useful information for focusing future efforts. In our domain, they indicate that further improvements in the non-understanding recovery rate are likely to translate into significant increases in task success, especially for the non-native user population, where 26.3% of the turns are non-understandings and the recovery rate is only 39.3%.

5 Performance of Non-understanding Recovery Strategies

We now turn our attention to the third question: ***how do the ten strategies compare with each other in terms of recovery performance?***

We computed the non-understanding recovery rate (as defined in the previous section) for each of the ten recovery strategies. The analysis is again performed only using the data collected in the *control* condition of our experiment. In this condition, the recovery strategies were engaged in an uninformed (random) fashion, and therefore they were on an equal footing. Figure 6 illustrates the resulting performance of each strategy, and the 95% confidence intervals for these estimates.

An overall analysis of variance for binary response variables (logistic ANOVA) revealed that there are statistically significant differences between the mean recovery rates of the 10 strategies (p=0.000035). Next, we used logistic ANOVAs to compare each pair of strategies individually. In each of these ANOVAs, we added the *nativeness* indicator as a factor in the model (since performance varies considerably between native and non-native users). The results are illustrated in Table 4, where each cell contains the ratio of the recovery rates between the strategies in the corresponding row and column. The resulting p-values (corresponding to the effect of strategy on recovery rate, when accounting for nativeness) were corrected for multiple comparisons using the false-discovery-rate method (Benjamini and Hochberg, 1995). This method allows us to compute the expected rate of false detections among the detected significant differences. The false-discovery-rate (FDR) for each result is illustrated by the shade of gray. For instance, we expect that 5% of the 10 cells with FDR=0.05 are actually not significant differences. While significant differences cannot be established for every strategy pair, the detected differences allow us to identify a partial ordering.

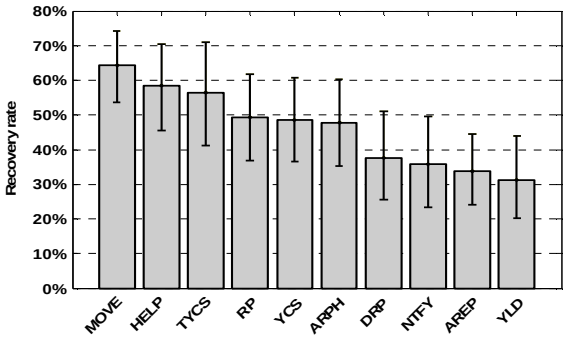

Figure 6. Individual strategy recovery rate

			MOVE	HELP	TYCS	RP	YCS	ARPH	DRP	NTFY	AREP	YLD
MoveOn	MOVE	64.4%	-	1.10	1.14	1.31	1.33	1.35	1.71	1.80	1.91	2.06
FullHelp	HELP	58.5%	-	-	1.03	1.19	1.20	1.22	1.55	1.64	1.73	1.87
TerseYouCanSay	TYCS	56.5%	-	-	-	1.15	1.16	1.18	1.50	1.58	1.68	1.81
Reprompt	RP	49.2%	-	-	-	-	1.01	1.03	1.31	1.38	1.46	1.58
YouCanSay	YCS	48.6%	-			-		1.02	1.29	1.36	1.44	1.55
AskRephrase	ARPH	48.6%	-	-	-	-	-	-	1.27	1.34	1.42	1.53
DetailedReprompt	DRP	37.7%	-	-	-	-	-	-	-	1.06	1.12	1.21
Notify	NTFY	35.7%	-	-	-	-	-	-	-	-	1.06	1.14
AskRepeat	AREP	33.7%	-	-	-	-	-	-	-	-	-	1.08
Yield	YLD	31.2%	-	-	-	-	-	-	-	-	-	-

Table 4. Comparison of non-understanding recovery rates; the cells show the ratio of the non-understanding recovery rate between the strategy in the corresponding row and column; the shading indicates the false-discovery-rate level (FDR=0.15 FDR=0.10 FDR=0.05)

The *MoveOn*, *Help* and *TerseYouCanSay* strategies occupy the top 3 positions, with no statistically significant differences detectable between them. In retrospect, this result is not surprising. A number of studies (Swerts et al., 2000; Rotaru and Litman, 2005) have shown that once an error has occurred, the likelihood of having an error in the next turn is significantly increased (our data also confirms this result). As we go deeper into a spiral of errors, patience runs out, frustration is likely to increase, and the acoustic and language mismatches are likely to become more pronounced. Moreover, the fact that there was a non-understanding in the first place indicates that the system is in a difficult position in terms of decoding the current user intention. When the system abandons the current question and attempts to solve the problem by using a different dialog plan, these effects are likely to be attenuated, and chances of correct understanding become higher. Similarly, when the system provides help including sample responses for the current question, the users might find better ways (from a system's perspective) to express their goals, or they might find out about other available options for continuing the dialog from this point.

The high performance of the *MoveOn* strategy is consistent with prior evidence from a wizard-of-oz study of error handling strategies (Skantze, 2003). Skantze's study has revealed that, unlike most spoken dialog systems, human wizards often did ***not*** signal the non-understandings to the user when they occurred. Instead, they asked different task-related questions to advance the dialog. This strategy generally led to a speedier recovery. In the RoomLine system, the MoveOn strategy implements this idea in practice, and the observed performance confirms the prior evidence from Skantze's study. Although not surprising, we do find this result very interesting, as it points towards a road less traveled in spoken dialog system design: when non-understandings happen, instead of trying to repair the current problem, use an alternative dialog plan to advance the task.

The next three strategies – *Reprompt*, *YouCanSay* and *AskRephrase*, form a second tier, all having a statistically better recovery rate than the last 4 strategies. Finally, no significant differences could be detected in terms of recovery rate between the last four strategies: *DetailedReprompt*, *Notify*, *AskRepeat* and *Yield*.

6 User Responses to Non-understanding Recovery Strategies

We now move on to the fourth question: ***what are the relationships between each strategy and subsequent user behaviors, and which behaviors are more likely to lead to successful recovery*.** Like before, the analysis is based on data from the *control* condition, where the strategies were engaged in an uninformed fashion.

To perform this analysis, we annotated each user turn that followed a non-understanding according to a tagging scheme for error segments introduced by Shin (2002), and subsequently used by others (Choularton and Dale, 2004; Raux et al, 2005). Like Choularton and Dale (2004), we used an abbreviated version of the original scheme, containing 5 labels: *repeat* – when the user is repeats the previous utterance identically, *rephrase* – when the user rephrases the same semantic content in a different lexical manner, *change* – when the user changes the semantic concepts with respect to the previous utterance, *contradict* – when the user contradicts the system, often as a barge-in and *other* – subsumes response types which do not fall in any of the previous categories (e.g. hang-ups, timeouts, etc.)

Figure 7 shows the overall distribution of user response types in our dataset. As a reference, we also show the user response type distributions found by Shin in an analysis of the Communicator corpus, and Choularton and Dale in an analysis of a deployed system for ordering pizza. Note however that a direct comparison

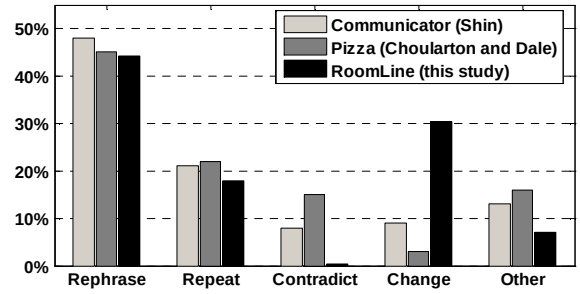

Figure 7. Distribution of user response types

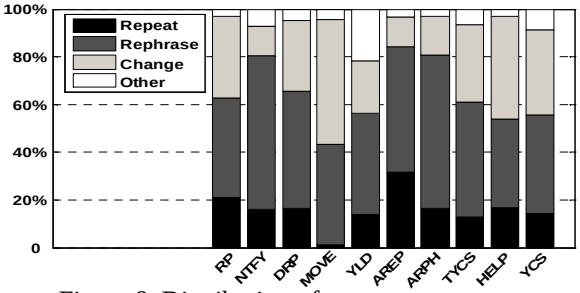

Figure 8. Distribution of user response types by non-understanding recovery strategy

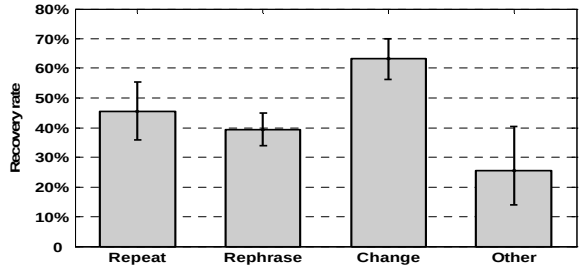

Figure 9. Recovery rate for different user response types

between these experiments is not valid since we only considered the user responses which followed a non-understanding (as opposed to throughout any error segment). The distribution of user response types we observed is nonetheless similar to previous studies. When faced with non-understandings, users tend to rephrase (~45%) more than repeat (~20%). A notable difference in the distribution appears between the *change* and *contradict* user response types. The fact that we only considered turns following non-understandings potentially explains the absence of *contradicts* (which happen mostly when a system misunderstands), while the large number of *change* responses is introduced by the *MoveOn* strategy - see Figure 8 and also additional plots available on-line (Bohus, 2005). While in Shin's study of the Communicator data a lot of *change* responses occurred as users were changing their travel plans to go around weaknesses in the system, this is not the case in this data. Participants in our study were compensated according to the number of scenarios they managed to complete *successfully*, and the *change* responses represent valid contributions to the dialog, within the confines of the given scenarios.

Next, we analyzed the impact of strategies on user response types. The results are presented in Figure 8. An auxiliary three dimensional representation of the strategies in the space of user response types is available online (Bohus, 2005). The results indicate that *AskRepeat* leads to the largest number of *repeat* responses (31%); the *MoveOn* strategy leads to the largest number of *change* responses (52%); the *AskRephrase* and *Notify* strategies lead to the largest number of *rephrase* responses (64%). While there is clearly an effect of strategy on user response types, the numbers shown above are not extremely large. Under the assumption that certain types of user responses are more desirable in certain circumstances, these results raise the question of whether the user response types can be controlled even more, for instance by using a more aggressive prompting style (e.g. "Could you repeat what you just said?" instead of "Can you please repeat that?")

Finally, we analyzed which type of user responses are more likely to lead to recovery. Figure 9 shows the recovery rate for each user response type. The best recovery performance is attained on *change* responses (63%). Together with the large number of *change* responses on the *MoveOn* and help strategies, this result corroborates the high performance of these strategies, and the discussion from the Section 5. Somewhat surprisingly, we were not able to establish a statistically significant difference between the recovery rates of user *repeat* and *rephrase* responses. In this respect, our results conflict with prior studies which have shown that user rephrases are better recognized and more likely to lead to recovery (Goldberg et al, 2003). Moreover, the same analysis performed on the sessions collected in the *wizard* condition (recall that in this case a human wizard decided which strategy should be engaged to recover) shows that in that case *repeat* responses were actually significantly better recognized than *rephrase* responses. Briefly, we believe this last result is explained by the fact that the wizard made intensive use of the *AskRepeat* strategy, *when this strategy was appropriate*; this in turn boosted the overall number as well as recovery performance of *repeat* responses.

Given these observations, we conclude this section on a cautionary note: while informative, results regarding the performance of various strategies and user responses do not necessarily generalize across domains. The success of various types of user responses can be strongly influenced by a number of factors such as the nature of the task, the user population, as well as the policy used to engage the strategies. We believe that the solution for successful recovery lies in endowing spoken dialog systems with the capacity to dynamically adjust their error handling behaviors to the specific characteristics of the domains in which they operate.

7 The Effect of Recovery Policy on Performance: Wizard versus Uninformed

So far we have concentrated our attention on the function and performance of individual recovery strategies. In the two remaining sections we will shift our focus to the non-understanding *recovery policy*. The *recovery policy* describes which strategy should be used in each situation.

Ultimately, our goal is to endow spoken dialog systems with the ability to automatically learn good recovery policies from their own experience. Our starting point is the hypothesis that the performance of various recovery strategies can be improved by engaging them at the right time, i.e. by using a good recovery policy. For example, asking the user to repeat is not a good course of action if the non-understanding was the result of an out-of-grammar utterance. In contrast, if the non-understanding was caused by a transient noise (e.g. a door slam), asking the user to repeat is probably more likely to succeed.

As a first step, we therefore wanted to confirm this hypothesis: ***can dialog performance be improved by using a better, more informed policy for engaging non-understanding recovery strategies?*** Its validity is not as obvious as it might seem. The performance of the error recovery process is a product of both the set of available strategies and the policy used to engage them. If the set of strategies does not provide good coverage for the types of problems we encounter, a good policy will fail to significantly increase performance. Should this be the case, our efforts would probably be better focused on developing more (and different) recovery strategies, rather than trying to learn a better policy.

To find an answer for the question raised above, we compared the performance of the wizard's recovery policy against the performance of the uninformed policy. Recall that the wizard had access to more information than a system would have at runtime, and therefore the detection of a performance gap between the policies does not prove that the wizard's policy is also attainable for a system; it only proves that a better policy exists (see discussion in subsection 2.1). We start by describing the dialog performance metrics we used in the comparison in subsection 7.1, and we present the results of the comparison in subsection 7.2. Finally, in subsection 7.3 we analyze the effect of the wizard policy on the performance of the individual non-understanding recovery strategies.

7.1 Performance Metrics

To evaluate global dialog performance we used two metrics: ***task success*** and ***user satisfaction***. Task success was defined as a binary variable for each of the 10 scenarios performed by a user. User satisfaction was expressed on a 1-7 Likert scale, and was elicited through a post-experiment questionnaire. The user satisfaction score corresponds therefore to the overall experience the user had with the system.

Apart from global dialog performance, we also wanted to assess the impact of the wizard policy on local non-understanding recovery performance. To our knowledge no traditional, well-established metrics exist in the community for performing this type of evaluation. We therefore constructed a number of metrics which we describe below. Each of these metrics evaluates various characteristics of the user response following the system's attempt to recover from a non-understanding.

The first metric, which we have already introduced in Section 4, was ***recovery rate***. To compute this metric, we simply look at whether the next user turn following a system attempt to recover is correctly understood or not. If the next turn is correctly understood (i.e. it is not a misunderstanding and it is not a non-understanding), then we say that the system has successfully recovered. *Average recovery rate* is then simply defined as the number of successful recoveries with respect to the total number of attempts to recover. The underlying variable in this metric is binary - the next turn is either correctly understood or not. The metric therefore does not take into account the magnitude or costs of potential errors. Nevertheless, this metric provides a first order estimate of recovery performance and (because of low variance) is especially useful when we have only a small number of samples to evaluate from.

A second metric we considered was ***recovery word-error-rate***. Instead of looking at whether the next turn is correctly understood or not, we compute and average the word-error-rate for the user turns following non-understanding recovery attempts. This metric captures in more detail the magnitude of the speech recognition errors in the user responses. However, in a spoken dialog system we are interested in the correctness of concepts acquired by the system rather than the correctness of the recognition process per se.

The third metric we used, ***recovery concept utility***, operates at the concept level. This metric takes into account the number of concepts that are correctly and incorrectly acquired by the system, as well as their relative utilities. The metric is computed as follows:

$$\boldsymbol{CU = Util_{CC} \cdot CC + Util_{IC} \cdot IC}$$

where ***CC*** is the number of concepts that are correctly acquired by the system from the user's response, and ***IC*** is the number concepts that are incorrectly acquired from that turn. $\boldsymbol{Util_{CC}}$ and $\boldsymbol{Util_{IC}}$ are weighting factors for the correctly and incorrectly acquired concepts and are obtained through a logistic regression model which relates the average number of correctly and incorrectly acquired concepts per turn to overall task success. A model constructed with in-domain data showed that

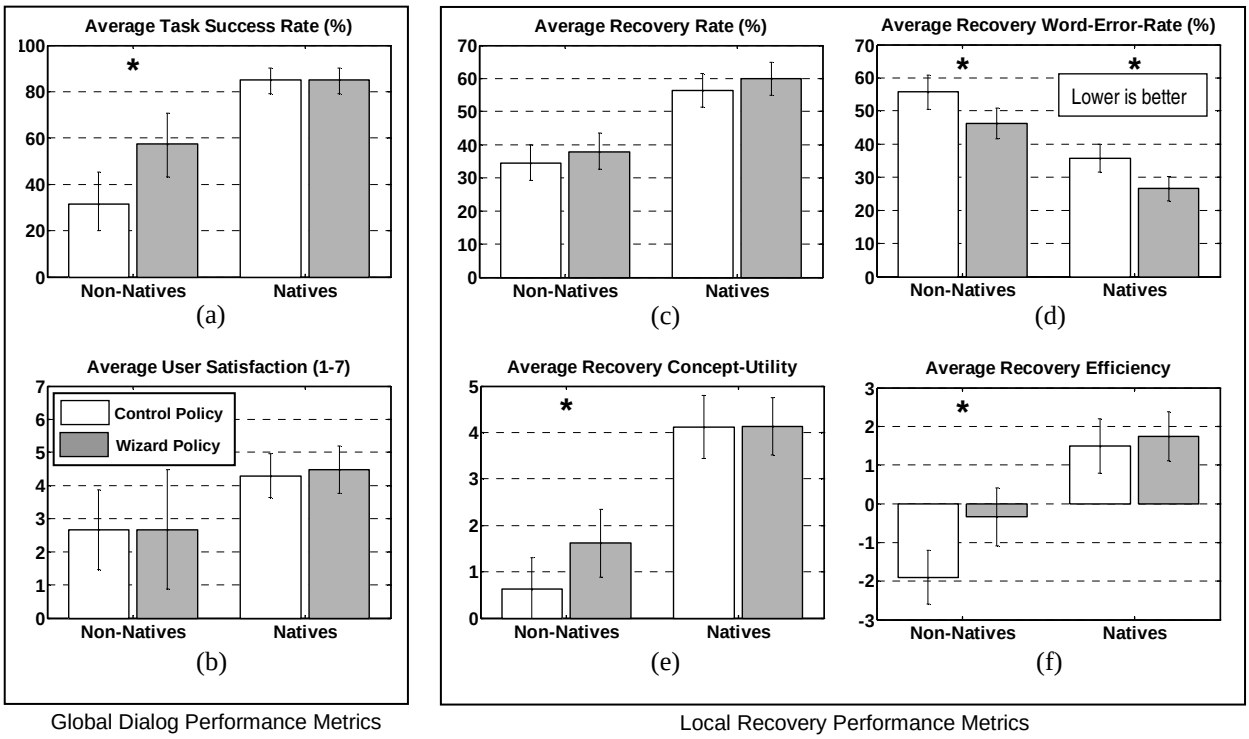

Figure 10. Performance comparison between the wizard and the uninformed recovery policy (* marks a statistically significant difference at $p < 0.05$)

Metric		Overall	Wizard vs Uninformed	Wizard vs Uninformed (only natives)	Wizard vs Uninformed (only non-natives)
Task Success (%)	(a)	75.1	78.5 ≈ 71.7	85.2 ≈ 85.2	57.4 > 31.6
User Satisfaction (1-7)	(b)	3.93	3.87 ≈ 4.00	4.29 ≈ 4.47	2.67 ≈ 2.67
Recovery rate (%)	(c)	48.7	50.1 ≈ 46.5	61.0 ≈ 56.4	37.9 ≈ 34.4
Recovery word-error-rate (%)	(d)	38.9	35.4 < 44.5	26.6 < 35.7	46.4 < 55.7
Recovery concept utility	(e)	2.80	3.01 ≈ 2.58	4.13 ≈ 4.12	1.62 > 0.63
Recovery efficiency	(f)	0.41	0.81 > 0.00	1.74 ≈ 1.50	-0.34 > -1.90

Table 5. Performance comparison between the wizard and the uninformed recovery policy (shaded cells mark differences that are significant at $p < 0.05$)

$\boldsymbol{Util_{CC}}$ = +7.81, and $\boldsymbol{Util_{IC}}$ = -7.19. For the interested reader, the methodology for deriving these costs is described in more detail in (Bohus and Rudnicky, 2005). Because it takes the domain-specific costs for correct and incorrect concepts into account, we consider this metric more appropriate than the traditional concept-error-rate.

Finally, the last metric we considered was ***recovery efficiency***. This metric goes one step further than the recovery concept utility, and also normalizes for the amount of time spent by the system during the recovery strategy. The motivation behind this metric is that some recovery strategies use shorter prompts than others, and therefore might succeed (or fail) faster. To normalize for the amount of time spent during recovery, we compute the number of concepts (correct and incorrect) we would expect the system to acquire on average during that time interval. We then subtract these numbers from the number of correct and incorrect concepts we did actually acquire in the next user turn. The formula for this metric is:

$$\boldsymbol{RE = Util_{CC} \cdot (CC - t \cdot rcc) + Util_{IC} \cdot (IC - t \cdot ric)}$$

where $\boldsymbol{t}$ is the time elapsed between the original non-understanding and the next user turn, and $\boldsymbol{rcc}$ (and $\boldsymbol{ric}$) are the average rates (per second) of acquiring correct (and incorrect) concepts during non-understanding recovery segments. In other words, during the amount of time $\boldsymbol{t}$ the system spent in its attempt to recover, we would expect to obtain on average $\boldsymbol{t \cdot rcc}$ correct concepts

and ***t·ric*** incorrect concepts. We subtract these from the actual number of correct (***CC***) and incorrect (***IC***) concepts we obtained in the user response, and then we take the corresponding utilities into account.

7.2 Results

The results of the comparison are shown in Table 5 and illustrated in Figure 10 (a)-(f). Since performance varies considerably between the native and non-native users, we present the breakdown of the differences in these two populations. In Table 5, the second column shows the overall performance (both groups together); the third column shows the overall differences between the *wizard* and the *control* conditions, while columns 4 and 5 show the differences between conditions within the native and non-native populations. The shaded cells mark differences that are statistically significant at a p-value smaller than 0.05. To test for statistical significance we used t-tests when comparing proportions (e.g. task success or recovery rate), and non-parametric Mann-Whitney U-tests for the other continuous-valued metrics (their values are not normally distributed).

As Figure 10 and Table 5 illustrate, an overall pattern emerges. The wizard policy does indeed lead to statistically significant performance improvements on a number of metrics, but the improvements appear mostly within the non-native population, i.e. in the group of users that had more difficulties using the system.

For instance, while no task success improvement can be detected for native users, there is a large task success improvement for non-native users (see Figure 10-a). The average task success rate grows from 31.6% in the *control* condition to 57.4% in the *wizard* condition. This increase bridges half of the original performance gap between native and non-native users in the *control* condition. Despite this increase in task success rate, no statistically significant differences can be detected with respect to user satisfaction (Figure 10-b); the small number of samples we have (one per user) and the large variance of this metric lead to wide confidence bounds on the mean estimates and preclude a reliable comparison. Nevertheless, the same trend of larger, statistically significant improvements for the non-native users is observed again on the local recovery performance metrics (Figure 10.c-f). Statistically significant improvements can be detected in the non-native population for three of these metrics: recovery word-error-rate, recovery concept utility, and recovery efficiency.

We believe the explanation for the observed result lies in the simple fact that it is easier to improve performance when performance is low (in our case, for the non-native users). This result also confirms our conjecture from Section 4: improvements in non-understanding recovery performance do indeed translate into significant increases in task success for the non-native population.

7.3 Effect of Policy on Individual Recovery Strategy Performance

Next, we analyzed the effect of the policy on the performance of individual recovery strategies. Our original hypothesis was that, if the strategies are engaged "at the right time", their performance would improve.

Figure 11 shows the number of times each non-understanding recovery strategy was engaged by the wizard. Figure 12 shows the *recovery rate* for each of the ten strategies, under the two different conditions. We were able to establish a statistically significant difference ($p=0.0023$, or $p=0.023$ Bonferroni corrected for multiple comparisons (Savin, 1980)) only for the *AskRepeat* strategy. *AskRepeat* is however the strategy most often engaged by the wizard. While this strategy ranked 9th when engaged in an uninformed fashion, its performance improved considerably from 33.7% to 53.0% under the wizard policy and is on par with the other top-performing strategies such as giving help (*TerseYouCanSay* and *Help*) or advancing the task by asking a different question (*MoveOn*). The same improvement in the *AskRepeat* strategy was also detected on the other three recovery performance metrics.

This result shows that strategy performance can indeed be improved by the use of a better recovery policy. At the same time, the lack of detectable differences in

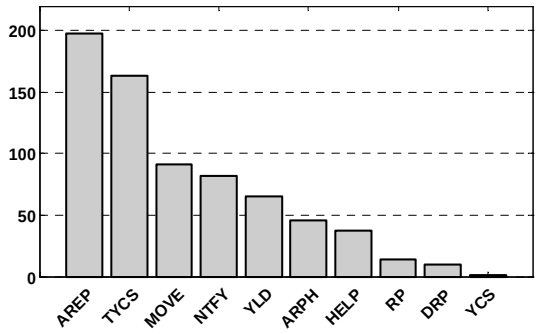

Figure 11. Number of times each strategy was engaged by the wizard

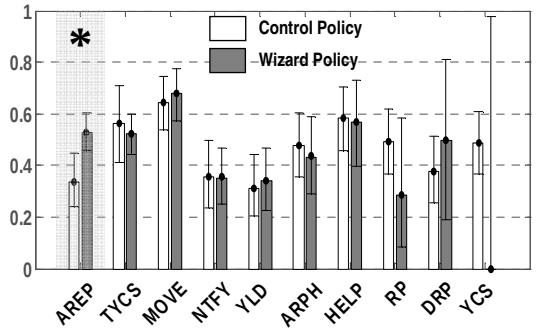

Figure 12. Impact of policy on individual strategy performance

the other strategies is somewhat disappointing. In retrospect, this result might be explained by the fact that the decision task the wizard had to perform was quite difficult, even with access to the full audio signal. To maintain the illusion that users were interacting with an autonomous system, the wizard had to choose one of ten recovery strategies in a very short time interval: 1 to 2 seconds. This selection task is easier for some of the strategies than for others. Furthermore, a number of strategies, such as *YouCanSay*, *Reprompt*, and *DetailedReprompt*, were very rarely engaged by the wizard and as a result the confidence intervals on their performance estimates are very wide, and preclude accurate comparisons.

8 Towards Learning a Recovery Policy

In the previous section we have established that significant improvements in performance can be achieved by using a better policy with the current set of strategies. In this section we present a set of preliminary results on the problem of ***learning such a policy from data***.

We take a simple, decision-theoretic approach. First, we learn to predict the likelihood of success for each non-understanding recovery strategy from features available at runtime. Then, to implement a policy, we compute the expected utility for each strategy (taking into account the probabilities and costs for success and failure), and select the strategy with the maximum expected utility.

In subsection 8.1, we describe the construction of predictors for the likelihood of success of each strategy. Next, in subsection 8.2 we discuss two recovery policies based on these predictors.

8.1 Predicting the Likelihood of Success for Non-understanding Recovery Strategies

We use logistic regression models to develop runtime predictors for the likelihood of success of each non-understanding recovery strategy.

Data. As training data, we use the turns in which a non-understanding occurred and the strategy we are interested in was engaged. The training target value is the success or failure of the strategy in that particular case. Success is defined as "the next user turn is correctly understood by the system".

Note that for learning we only use data collected in the *control* condition, where the non-understanding recovery strategies were engaged in an uninformed fashion. The wizard policy might introduce a potential bias in the distribution of features, which can negatively affect learning and generalization. For instance, if the wizard never used the *AskRepeat* strategy when the number of words in the original non-understood utterance was very large, we will never encounter that set of circumstances, or see how the *AskRepeat* strategy behaves under those conditions. In this case, the distribution of the *number_of_words* feature might be skewed towards small values, and that might negatively affect the learning process.

Given the relatively large number of different recovery strategies in the system (10), the number of available instances to learn from is fairly small – about 60 to 70 invocations per strategy. The small number of samples further complicates an already difficult learning problem, since we face a relatively high risk of over-fitting the training data.

Features. We identified a large number of features available at runtime which could carry information about the likelihood of success for various non-understanding recovery strategies. The features are collected from different levels of processing in the spoken dialog system. For instance, from the speech recognition level we collected various features characterizing the current non-understood utterance: the number of words, the signal and noise levels, the number of and proportion of words tagged as unconfident by the speech recognizer. Similarly, from the language understanding level we collected various features reflecting the quality of the parse. From the dialog management level we used information about the dialog state, as well as the history of the dialog up to that point (e.g. how many previous consecutive non-understandings we encountered, what was the average confidence score so far, etc.)

As a first measure to guard against over-fitting, we transformed continuous features into binary features by using a preset threshold. Furthermore, we eliminated features that had small class-conditional counts. The remaining set of features which was used in training is available as an online appendix (Bohus, 2005).

Models. Since we are interested in predicting the expected likelihood of success for each strategy (rather than a binary success or failure), we decided to use stepwise logistic regression models. These models are simple, easy to build and incorporate a mechanism for feature selection. Moreover, as opposed to a number of other discriminative classifiers, logistic regression provides good class posterior probability scores (e.g. estimates for the likelihood of success).

In stepwise logistic regression, features (variables) are added to the model one by one, as long as they increase the likelihood of the data. A feature is accepted in the model if it produces a data likelihood increase that is statistically significant with a p-value below a preset *P-accept*. At the same time, in each step features already in the model are tested for exclusion. A feature is rejected if the resulting model is not significantly worse, as determined by a preset *P-reject*. In our case, we set *P-accept=0.05* and *P-reject=0.30*. Finally, as a second preventive measure against over-fitting, we evaluated the model after each regression step using a

Strategy	# training instances	Average data log-likelihood			Hard error rate (%)		
		Majority Baseline	Leave-one-out perf.	Avg. log-lik. increase	Majority Baseline (error rate)	Leave-one-out perf. (error rate)	Relative reduction in error rate
DetailedReprompt	61	-0.6626	-0.6330	0.0296	37.7%	34.4%	8.7%
Reprompt	67	-0.6930	-0.6645	0.0286	49.2%	32.8%	33.3%
TerseYouCanSay	46	-0.6846	-0.6576	0.0270	43.5%	32.6%	25.0%
MoveOn	90	-0.6508	-0.6357	0.0151	35.6%	30.0%	15.6%
YouCanSay	70	-0.6927	-0.6729	0.0199	48.6%	34.3%	29.4%
AskRepeat	89	-0.6391	-0.6389	0.0002	33.7%	33.7%	0.0%
Help	65	-0.6788	-0.6776	0.0012	41.5%	46.1%	-11.1%
AskRephrase	67	-0.6921	-	-	47.8%	-	-
Notify	56	-0.6518	-	-	35.7%	-	-
Yield	64	-0.6211	-	-	31.2%	-	-

Table 6. Performance of success predictors for the 10 non-understanding recovery strategies

leave-one-out procedure and stopped adding features as soon as the average data likelihood in the leave-one-out evaluation decreased.

Results. We fitted ten step-wise logistic regression models, one for each strategy. The results are illustrated in Table 6. For 5 of the 10 strategies we can build models which perform better than a majority baseline, on both a soft (average log-likelihood) and hard (binary classification) error metric. For the last three models in Table 6 no features ever entered the regression. In this case the constructed predictors simply predict a probability of success equal to the majority baseline in the training data. In general, the performance of the individual predictors is not very good, but this is not surprising given the small number of training instances, the reduced number of features used, and difficulty of the prediction task (we are trying to predict in advance whether or not the next turn is correctly understood, without any information from that turn.)

8.2 Policies for Recovery

If we can predict for the likelihood of success of each non-understanding recovery strategy, a recovery policy is easy to construct: we simply choose the action with the maximum expected utility:

$$\Pi = \text{argmax}\ \{\ P_{SUCC}(S) \cdot U_{SUCC}(S) + P_{FAIL}(S) \cdot U_{FAIL}(S)\ \}$$

where $P_{SUCC}(S)$ is the estimated probability of success for strategy S, $P_{FAIL}(S) = 1 - P_{SUCC}(S)$ is the probability of failure, and $U_{SUCC}(S)$ and $U_{FAIL}(S)$ are the utilities of success and failure for strategy S.

We defined two policies. The first policy (***max-recovery-rate***) aims to maximize the recovery rate by choosing the strategy with the maximum likelihood of success. This is equivalent to using the values U_{SUCC}=1 and U_{FAIL}=0 as the utilities for success and failure. The second policy (***max-recovery-efficiency***) aims to maximize the recovery efficiency, as defined in subsection 7.1. In this case $U_{SUCC}(S)$ is the average recovery efficiency of strategy S when S was successful (i.e. the next turn is correctly understood), while $U_{FAIL}(S)$ is the average recovery efficiency of strategy S when S failed.

To obtain a preliminary estimate for the performance of these policies, we looked at what happened in the data from the *wizard* condition, when the wizard happened to make the same decisions as our learned policies would have made. Since the *MoveOn* strategy was not available at all points in the dialog, we eliminated it from the learned policies in this analysis (this is a simple way to avoid the policy deciding to engage the *MoveOn* strategy when it is not available). The results show that, within the subset of instances where the wizard made the same decision as the ***max-recovery-rate*** policy, the recovery rate performance was 69.8%. At the same time, the wizard's *overall* recovery rate (throughout the whole *wizard* dataset) was significantly lower – 50.1%; the overall recovery rate with the uninformed policy from the control group was 46.5%[1]. Similarly, on the instances where the wizard agreed with the ***max-recovery-efficiency*** policy, the recovery efficiency performance was 2.02, significantly larger than the overall wizard recovery efficiency (0.81), and the uninformed policy recovery efficiency (0.00).

While we view these results as promising, we would like to point out a potential problem in this type of evaluation. Given that both the wizard and the learned policy strive to maximize performance, the distribution of the subset of non-understandings where they agree might not be representative for the true distribution of non-understandings – these might be the cases where it's easier to tell which strategy should be used to recover. Ultimately, a new user study where the system runs with the learned policy is required in order to robustly evaluate its performance.

[1] If we also eliminate the *MoveOn* strategy from the assessment of the overall *wizard* and *control* performance, the numbers become 46.6% for the *wizard* condition and 44.0% for the *control* condition.

9 Conclusion

The work described in this paper is part of a larger research program (Bohus, 2004) aimed at endowing spoken dialog systems with better error handling capabilities. In an effort to shed more light on non-understandings, we performed an empirical analysis of these errors and ten associated recovery strategy, based on a corpus of dialogs collected with a mixed-initiative spoken dialog system for making conference room reservations.

An error source analysis has confirmed that a large number of non-understandings (and misunderstandings) can be blamed on speech recognition errors. Nevertheless, a significant number of non-understandings (~30%) stem from requests for inexistent application functionality, user corrections, and out-of-grammar expressions. At the same time, users are not always aware of the full functionality provided by the system. We believe these language-domain errors can be addressed by better steering the users into the application's space. In this vein, we plan to explore more carefully the design and use of *you-can-say* help messages. Currently these messages inform users about possible ways to answer the current question. In the future, we plan to investigate the possibility of providing information about other options available at this point in the dialog. This raises an interesting design issue, since the number of different options available to the user can be fairly large in a mixed-initiative spoken dialog system. A *targeted-help* approach (Gorrell et al., 2002) may provide a potential solution to this problem. A second path we intend to explore is issuing *you-can-say* messages preemptively (e.g. without waiting for a non-understanding to happen) if we can detect that the user belongs to a "problematic" population such as non-native speakers, first time users, etc.

We have also confirmed that non-understandings exert a negative impact on task success, and we have quantified this impact. Models discussed in Section 4 revealed that the effect of non-understandings on task success is marginal when the frequency of non-understandings is below 10-15%, but increases fast after that. Similarly, we found that the effect of the non-understanding recovery rate on performance is greatest when the recovery rate is below 60-70%, and smaller once we are above that limit. While the specific numbers might differ across applications and domains, we expect that the nature of the relationship remains similar. The type of analysis we presented in Section 4 can provide useful information for focusing future development efforts. In our domain, it indicates that improvements in the non-understanding recovery rate are likely to lead to significant increases in task success, especially for non-native users (where the non-understanding rate is relatively high, and the non-understanding recovery rate is relatively low).

Next, we compared the individual performance of the ten different recovery strategies. The results show that, when engaged without any prior knowledge, the best performing strategies in our domain are: (1) advancing the conversation by ignoring the non-understanding and trying an alternative dialog plan (*MoveOn*), and (2) providing help messages containing sample responses for the current system question. The high performance of the *MoveOn* strategy corroborates prior evidence from a wizard-of-oz study (Skantze, 2003) which showed that human operators often do not signal non-understandings, but rather try to advance the task by asking different questions. In the future, we plan to explore in more detail the potential uses of this strategy, as well as its pitfalls. Specific issues we plan to investigate include identifying more situations in which this strategy is applicable, studying the extent to which this strategy can be decoupled from the system's task, and developing more appropriate metrics for assessing its performance.

In the final part of this paper we shifted our attention to the recovery policy. We showed that a more informed policy for engaging the non-understanding recovery strategies (implemented by a human wizard) led to significant improvements in task success, as well as a number of other local performance metrics. The improvements occurred mostly within the non-native population, i.e. the group of users who had more difficulties in using the system.

Finally, we reported a set of preliminary results with respect to learning a recovery policy from data. We used features available at runtime from various levels in the dialog system to build predictors for the likelihood of success of each non-understanding recovery strategy. For five of the strategies, the learned predictors perform better than a majority baseline. Based on these predictors, we constructed two policies: one aims to maximize the recovery rate, the other aims to maximize the recovery efficiency. Preliminary estimates indicate that these policies are expected to outperform both the wizard and the uninformed policy. While these experiments were conducted with very few training instances and an empirical validation of the learned policy is still necessary, we find these results encouraging as they indicate the feasibility of using a learning approach for deriving a non-understanding recovery policy from data.

Acknowledgements

This work, part of the CALO project, was supported by DARPA grant NBCH-D-03-0010. 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. The authors would like to thank Antoine Raux, Valerie Ventura, Mihai Rotaru, the members of the "Dialogs on Dialogs" group, as well as the anonymous reviewers for

helpful suggestions and feedback. Last but not least, we would like to thank the workshop organizers and program committee for the opportunity to combine our two accepted submissions into a single longer paper format.

References

Benjamini, Y., and Hochberg Y. 1995. *Controlling the false discovery rate: a practical and powerful approach to multiple testing*. Journal of the Royal Statistics Society, B, 57, 289–300.

Bohus, D., Rudnicky, A., 2003. *RavenClaw: Dialogue Management Using Hierarchical Task Decomposition and an Expectation Agenda"*, in Proceedings of Eurospeech 2003, Geneva, Switzerland

Bohus, D., 2004. *Error Awareness and Recovery in Task-Oriented Spoken Dialog Systems*, Ph.D. Thesis Proposal, Carnegie Mellon University, Pittsburgh, USA.

Bohus, D., 2005. *Web page with auxiliary information for the investigation into non-understandings errors and recovery strategies*, www.cs.cmu.edu/~dbohus/nonu_investigation/

Bohus, D., Rudnicky, A., 2005. *A Principled Approach for Rejection Threshold Optimization in Spoken Dialog Systems*, submitted to Interspeech-2005, Lisbon, Portugal

Carpenter, P., Jin, C., Wilson, D., Zhang, R., Bohus, D., Rudnicky, A., 2001. *Is This Conversation on Track?*, in Proceedings of Eurospeech-2001, Aalborg, Denmark

Choularton, S., Dale, R., 2004. *User Responses to Speech Recognition Errors: Consistency in Behavior across Domains*, in Proceedings of SST-2004.

Choularton, S., 2005. *Investigating the Acoustic Sources of Speech Recognition Errors*, submitted to Interspeech-2005, Lisbon, Portugal

Clark, H.H. 1996. *Using Language*. Cambridge Univ. Press.

Goldberg, J., Ostendorf, M., Kirchhoff, K., 2003. *The Impact of Response Wording in Error Correction Subdialogs*, in Proc. of ISCA Workshop on Error Handling in Spoken Dialogue Systems, Chateau d'Oex Vaud, Switzerland, 2003.

Gorrell, G., Lewin, I., Rayner, M., 2002. *Adding Intelligent Help to a Mixed Initiative Spoken Dialogue System"*, in Proceedings of ICSLP-2002, Denver, CO.

Hone, K., Graham. R., 2000. *Towards a tool for the Subjective Assessment of Speech System Interfaces (SASSI)*, in Journal of Natural Language Engineering, 6:287-303, 2000, Cambridge University.

Krahmer, E., Swerts, M., Theune, M., Weegels, M., 1999. *Error Detection in Human-Machine Interaction*, Speaking. From Intention to Articulation, MIT Press, Cambridge, Massachusetts, USA

Litman, D., Hirschberg, J., Swerts, M., 2000. *Predicting Automatic Speech Recognition Performance Using Prosodic Cues*, in Proc. of NAACL-2000, pp. 218-225, Seattle, USA

Myers, R., Montgomery, D., Vining, G., 2001. *Generalized Linear Models: With Applications in Engineering and the Sciences*, Wiley-Interscience, ISBN 0471355739

Paek, T., Horvitz, E. 2000. *Conversation as Action Under Uncertainty*, in Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence

Raux, A., Langner, B., Bohus., D., Black, A.W., Eskenazi, M., 2005. *Let's Go Public! Taking a Spoken Dialog System to the Real World*, submitted to Interspeech-2005, Lisbon, Portugal

RoomLine, 2003 –http://www.cs.cmu.edu/~dbohus/RoomLine

Rotaru, M., Litman, D., 2005. *Interactions between Speech Recognition Problems and User Emotions*, submitted to Interspeech-2005, Lisbon, Portugal

San-Segundo, R., Pellom, B., Ward, W., 2000. *Confidence Measure for Dialogue Management in the CU Communicator System*, in Proc. of ICASSP-2000, Istanbul, Turkey

Sanders, G., Le, A., Garofolo, J., 2002. *Effects of Word Error Rate in the DARPA Communicator Data During 2000 and 2001*, in Proceedings of ICSLP-2002, Denver, CO, USA

Savin, N E, 1980. *The Bonferroni and the Scheffe Multiple Comparison Procedures*, in Review of Economic Studies, Blackwell Publishing, vol. 47(1), pages 255-73.

Shin, J., Narayanan, S., 2002. *Analysis of User Behavior under Error Conditions in Spoken Dialogs*, in Proceedings of ICSLP-2002, Denver, CO, USA

Skantze, G., 2003. *Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems*, in Proc. of ISCA Workshop on Error Handling in Spoken Dialogue Systems, Chateau d'Oex Vaud, Switzerland, 2003.

Swerts, M., Litman, D., Hirschberg, J., 2000. *Corrections in Spoken Dialogue Systems*, in Proc. of the 6th International Conference of Spoken Language Processing *(ICSLP-2000)*, Beijing, China, October 2000.

Theta, 2004. *Theta: A Small Footprint Text-to-Speech Synthesizer*, Cepstral LLC, Pittsburgh, PA, 2004, www.cepstral.com

Tomko, S., 2004. *Improving User Interaction with Spoken Dialog Systems via Shaping*, Ph.D. Thesis Proposal, Carnegie Mellon University, Pittsburgh, USA

Walker, M., Passonneau, R., Boland, J., 2001. *Quant-itative and Qualitative Evaluation of the DARPA Communicator Spoken Dialogue Systems*, in Proceedings of ACL'2001.

Walker, M., Litman, D., Kamm, C., and Abella, A. 2002. *Evaluating Spoken Dialogue Systems with PARADISE: Two Case Studies*, in Computer Speech and Language, 12-3

Ward, W., Issar, S., 1994. *Recent Improvements in the CMU Spoken Language Understanding System*, in Proceedings of the ARPA Human Language Technologies Workshop, March 1994, 213-216