Learning Decision Models in Spoken Dialogue Systems via User Simulation

Ed Filisko and **Stephanie Seneff**
MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Cambridge, Massachusetts 02139, USA
{filisko,seneff}@csail.mit.edu

Abstract

This paper describes a set of experiments designed to explore the utility of simulated dialogues and automatic rule induction in spoken dialogue systems. The experiments were conducted within a flight domain task, where the user supplies source, destination, and date to the system. The system was configured to support explicitly about 500 large cities; any other cities could only be recovered through a spell-mode subdialogue. Two specific problems were identified: the *conflict problem*, and the *compliance problem*. A RIPPER-based rule induction algorithm was applied to data from user simulation runs, and the resulting system was compared against a manually developed baseline system. The learned rules performed significantly better than the manual ones for a number of different measures of success, for both simulations and real user dialogues.

Introduction

Spoken dialogue systems are a natural means to interact with on-line information sources. Although the effectiveness of such systems has improved significantly over the past few years, a critical barrier to widespread deployment remains in the form of communication breakdown at strategic points in the dialogue. Such breakdown often occurs when the user is trying to convey a named entity from a large or open vocabulary set. For example, there is no easy way to inform a user about which cities a weather or flight domain knows.

While speech recognition has been steadily improving over time, it is still only able to provide best hypotheses of what a user has said, especially if the user's intended value is unknown. Confidence scores and unknown word models can help; however, it is still possible for a completely incorrect hypothesis to have high confidence and vice versa, especially when an unknown word sounds similar to a known word. For example, the unknown city, *Chaffee*, may be misrecognized as the known city, *Chelsea*.

One problem with confidence thresholds is that they are often based on simple heuristics and they remain static throughout the dialogue. Confidence may be a better or worse indicator depending on the specific user, the dialogue context, and the noise level. One nice approach to alleviating this problem has been elegantly addressed in work by Bohus and Rudnicky (2005), in which state-specific rejection thresholds are optimized over the course of a dialogue.

For a low-confidence attribute, an alternative to rejection is to initiate specific recovery strategies, such as a speak-and-spell subdialogue (Filisko & Seneff 2005). While a developer can specify rules for which action to take in a given situation, a significant amount of manual effort and time is typically required. A more efficient alternative is to automatically learn a set of rules, thus reducing the manual effort and perhaps even improving upon the manual rule set. User simulation facilitates this task by allowing a large number of simulated dialogues to be generated, providing both example training material and an automatic evaluation procedure, independent of real user data collection.

In this paper, two specific situations are explored, for which manually specifying decision rules is difficult: the *conflict problem*, in which the system must decide between two conflicting hypotheses for the same attribute, and the *compliance problem*, in which the system must predict whether a user has been compliant with a spelling request.

The next section describes the conflict problem and two approaches, manual rules and learned rules, for determining which action to take in that situation. A performance analysis of the models with a simulated user is provided. The compliance problem is then described, along with the resulting learned model. In order to test the effectiveness of the learned models, a telephone-based dialogue system was introduced to real users. Following an analysis of the resulting real user data, the paper concludes with a summary and discussion of future plans.

The Conflict Problem

The *conflict problem* is defined as a situation where a new value appears to be in conflict with a previously assigned value for a particular attribute. How should the system react? Should the new value be ignored, implicitly or explicitly confirmed, or substantiated by a spelling? Figure 1 shows a dialogue in progress between a simulated user (U) and a flight reservation system (S), in which this situation occurs. The system has already established that the source city is *Asheville*. A problem is encountered when the simulated user provides a date in U1. The system hypothesizes that the

S0:	*To Spring Mount. Can you provide a date?*
U1:	Well, let's see, on fifteen June.
S1:	*From Vienna. Can you try saying the date again?*
U2:	Okay, I want to leave Asheville on fifteen June.
S2:	*Did you say on June fifteenth?*
U3:	Yes.
S3:	*Okay from Vienna to Spring Mount on June 15th...*
U4:	I would like to leave from Asheville.
S4:	*Okay from Asheville to Spring Mount on June 15th...*

Figure 1: Dialogue between a simulated user (U) and a flight reservation system (S). The system hypothesizes *Vienna* from the user's utterance (U1), implicitly confirming this incorrect city (S1).

user said *Vienna*, with high confidence. The conflicting and incorrect source city, *Vienna*, is then implicitly confirmed in S1; ideally, it should have been ignored. Incidentally, no date was hypothesized. Consequently, in U2, the user must again provide the date, but must also correct the source city. The user repeats *Asheville* until the system finally correctly recognizes the city name and implicitly confirms it in S4.

A Manual Solution

One way to address the conflict problem is to manually specify a rule set, dictating how to react to the new value, conditioned on dialogue history. An initial rule set is often basic and inadequate; however, the developer typically modifies and refines the rules as the system is used. One drawback of such a method is the amount of time required to devise and hand-tune the rules. In addition, the developer is likely unable to objectively handle all possible dialogue situations.

In our experiments, a baseline system was first developed using a small set of ordered rules to handle the situation where a new value is hypothesized for a previously assigned source or destination. If the prior belief is grounded, because either the user confirmed it or it had been spelled, then the new value is ignored, unless its confidence score is sufficiently high. Once the decision is made to acknowledge the new value, it is subjected to the same rules that would apply if it were being introduced for the first time – the system consults a spectrum of static, manually specified confidence thresholds to determine whether to invoke explicit or implicit confirmation, or to initiate a spell-mode subdialogue.

A Learned Solution

An alternative approach to addressing the conflict problem is to employ a learning mechanism. Reinforcement learning has commonly been used in dialogue management to learn optimal strategies (Singh *et al.* 2002; Henderson, Lemon, & Georgila 2005), while supervised learning has been used to predict problematic dialogues (Litman & Pan 2000; van den Bosch, Krahmer, & Swerts 2001; Walker *et al.* 2002), as well as user corrections (Hirschberg, Litman, & Swerts 2001). Supervised learning approaches typically learn from static corpora, as opposed to simulated dialogues. However, simulated dialogues can be useful for some purposes since they enable the generation of a large number of dialogues without the need for real users (Levin, Pieraccini, & Eckert 2000).

cur_gnd	Grounding of currently believed city
conf	Confidence score of new city name
num_nbest_values	No. unique cities for slot in N-best list
value_is_oov	Is new value unknown hypothesis?
nbest_dominance	No. slots with new value in N-best list
found_in_sas_list	Current belief was spelled by user

Table 1: A subset of the 16 features chosen to predict the "ideal" system response in the case of the conflict problem.

The method used here to learn a decision model is as follows: in a simulated dialogue, upon encountering the conflict problem, the system records a set of features from the N-best list and the dialogue state, along with an "ideal" action; that is, the response that should lead to the fewest number of turns to successfully complete the dialogue task. This action is determined from the current state of beliefs, including the user's true intentions, which are provided to the dialogue manager by the simulated user. Of course, this information is used only for the purpose of assessing the actions taken by the system; it does not influence which action the system actually takes. These data can then be used to produce a model, mapping a feature vector to an action.

The "ideal" actions are defined as follows:

1. **Ignore**: when the system's belief is the user's intended city
2. **Implicit Confirm**: when the newly hypothesized city is the user's intended city
3. **Explicit Confirm**: when the newly hypothesized city is *not* the user's intended city, but the user's intended city is a *known* city
4. **Request Spelling**: when the user's intended city is *unknown*

If a city is unknown, spelling is the only way for the system to acquire it. While unable to provide information on that city, the system could, for example, propose a nearby city or tell the user that it does not have information on that specific city. For *Explicit Confirm*, the system asks, for example, "Did you say Houston?". If the suggested city were incorrect, users have been observed to include the *correct* city name in their reply, such as, "No, I said Boston". The system would hopefully be able to correctly recognize the user's intended, *known* city if it were repeated.

In order to know the ideal action, the system must know the user's true intentions. When real users are involved, this information is only available following manual annotation by experts. User simulation, however, enables access to this knowledge, allowing the automatic assessment of its decisions as they are made.

Features Various features have been used by researchers in learning approaches to dialogue management, including prosodic cues (Hirschberg, Litman, & Swerts 2001) and discourse features (Higashinaka, Sudoh, & Nakano 2005).

A set of 16 features was utilized for this work, a subset of which is illustrated in Table 1. These features are based on information from the recognizer, as well as from the dialogue state. Several of them are derived from the recognizer's N-best list, including *num_nbest_values* and *nbest_dominance*. For example, if *num_nbest_values* is relatively high, it might suggest that the user's intended value is unknown and that a spelling request might be appropriate.

Some features are also based on information stored in the

```
2. (:nbest_dominance >= 0.722222)
   and (:nbest_slot_absence <= 0)
   and (:num_nbest_values <= 4)
   and (:first_occur_depth <= 0.0625)
   and (:conf <= 4766)
    => action = implicit (11.0/2.0)
5. (:cur_gnd = hi_conf)
   and (:nbest_oov_count >= 1)
    => action = spelling (104.0/18.0)
11. => action = ignore (551.0/50.0)
```

Figure 2: Three rules from the set learned by JRip to predict the "ideal" response to the conflict problem. The numbers in parentheses represent *coverage/errors* in the training data.

dialogue state. One example is *cur_gnd*, which represents the grounding of the system's current belief. It can assume values of *not confirmed*, *high confidence*, and *confirmed by user*. *Not confirmed* indicates that the value was recognized with a relatively low confidence. If a conflicting value were hypothesized in this case, it would seem more likely for the currently believed value to be overridden than if the value of *cur_gnd* were *confirmed by user* or *high confidence*.

The Learned Rules The training data were obtained by simulating 596 dialogues with a compliant user; that is, the user always provided a speak-and-spell utterance when requested. Most utterances were generated with a concatenative speech synthesizer, combining real and synthesized speech (Filisko & Seneff 2005), while any spelling utterances were generated by DECtalk. The recognizer had explicit knowledge of about 500 major cities. In 298 dialogues, the user asked about flights from a city *known* to the recognizer to a city *unknown* to the recognizer. In the remaining dialogues, the user asked about flights from an unknown to a known city. The 298 known cities were each used twice, and the 596 unknown cities, only once. The city names were randomly combined into unique pairs prior to the simulations.

The task of the simulated user was to get the system to correctly understand the source city and state, the destination city and state, and the date. The user's first utterance contained only the source and destination cities; the state and date were provided only when requested by the system.

The simulated dialogues gave rise to 890 instances of the conflict problem, for which the corresponding feature vector/ideal action pairs were recorded. These data were fed into JRip, an implementation of the RIPPER rule induction algorithm (Cohen 1995), provided through Weka, a publicly available, Java-based machine learning toolkit (Witten & Frank 2005). A RIPPER-based implementation was chosen since RIPPER is commonly used in supervised learning approaches to dialogue management (Litman & Pan 2000; Hirschberg, Litman, & Swerts 2001; van den Bosch, Krahmer, & Swerts 2001; Walker *et al.* 2002).

Figure 2 shows three of the 11 learned rules, for which the actions are *implicit confirm*, *request spelling*, and *ignore*. The rules are traversed linearly and the action corresponding to the first matching rule is performed. All but two of the original 16 features were represented in the rules. The *cur_gnd* feature occurs in eight of the rules, indicating the importance of the current belief's grounding when considering whether to doubt its correctness.

	NumSucc	AvgTurns	AvgAttrAcc
Manual	33.2	7.8	86.4%
Learned	35.9	7.7	89.0%

Table 2: Results for the *Manual* and *Learned* models in number of successful dialogues (*NumSucc*), average turns per successful dialogue (*AvgTurns*), and average attribute accuracy (*AvgAttrAcc*). The values represent averages over ten runs.

Some of these rules demonstrate a degree of complexity not likely to arise from manual specification. Rule 2, for example, says that if at least 72.2% of the relevant slot values in the N-best list are the newly hypothesized value, none of the N-best hypotheses lack a value for the relevant slot, the N-best list contains at most four unique values for the relevant slot, the newly hypothesized value occurs in the top-best utterance, and the confidence of the new value is at most 4766, then the new value should be implicitly confirmed.

On the other hand, some learned rules are similar to manual rules. For instance, Rule 5 says that, if the current belief was acquired with high confidence and the N-best list contains at least one unknown word hypothesis for the relevant slot, then the system should request a spelling from the user.

Testing and Results

In order to compare the performance of the learned and manual models, 50 test scenarios, distinct from the training scenarios, were processed through each of the two system configurations. In each scenario, a compliant simulated user had to convey a known source, an unknown destination, and a date to the system. Each scenario was run ten times, and the results were averaged. The initial utterance for each scenario was identical, but the wording of subsequent utterances varied since they were randomly chosen from a set of templates, as described by Filisko and Seneff (2005).

Table 2 provides a comparison of the average results for both models. Each dialogue involved five attributes: source city and state, destination city and state, and date. For the learned model, the average number of successful dialogues was 35.9, as contrasted with 33.2 for the manual model, along with a significant increase in the average attribute accuracy ($p \leq .002$) and a slight, though not significant, decrease in the number of turns per successful dialogue.

A detailed analysis revealed how well each model was able to predict the ideal action in response to the conflict problem. Tables 3 and 4 provide confusion matrices for the action taken when the manual and learned models, respectively, were followed. Overall, the learned model significantly outperformed the manual model, with an accuracy of 76.1% compared to 61.7%.

Both models performed best in predicting the *ignore* action – the manual model was correct just over 71% of the time, while the learned model achieved nearly 85% accuracy. The models also predict the *ignore* action with similar frequencies – a little over 77% of the time. Since *ignore* is the default action for the learned rules, the learned model's superior performance is a consequence of the greater refinement and complexity of the rules preceding the default rule.

Reference→ *Prediction↓*	Ignore (64.0%)	Implicit (10.7%)	Explicit (3.4%)	Spelling (21.9%)
Ignore (77.4%)	38.6	3.9	1.8	10.0
Implicit (14.1%)	6.2	1.8	0.2	1.7
Explicit (2.1%)	0.0	0.7	0.0	0.8
Spelling (6.4%)	0.1	1.1	0.4	2.9

Table 3: Confusion matrix for decision made using the manual decision model. An overall accuracy of 61.7% was obtained. The values represent average number of decisions over ten runs.

Reference→ *Prediction↓*	Ignore (71.7%)	Implicit (9.5%)	Explicit (2.7%)	Spelling (16.1%)
Ignore (77.3%)	43.6	2.3	0.8	4.6
Implicit (8.6%)	2.7	1.9	0.0	1.1
Spelling (14.1%)	1.3	2.1	1.0	5.0

Table 4: Confusion matrix for decision made using the learned decision model. The overall accuracy was 76.1%. The values represent the average number of decisions over ten runs.

In the case of implicit confirmation, neither model performed very well (18% manual, 33% learned). Both models tended to predict implicit confirmation when the ideal action was, in fact, to *ignore* the newly hypothesized value. This confusion can be quite punishing since implicitly confirming a value, when it is *not* correct, places a burden on the user to correct the value. This can be difficult and frustrating for the user, especially if the system is highly confident in the incorrect value.

Explicit confirmation was never predicted by the learned rules. While it did happen to be the ideal action in a very small number of cases, the manual model also never successfully made that prediction, so the conservative approach of the learned rules was probably appropriate.

The learned model requested significantly more spellings – 14.1% of its predictions, versus the manual model's 6.4%. Perhaps as a result of such aggressiveness, the learned model was less accurate (5.0/9.4=53%) than the manual model (2.9/4.5=64%) when predicting a spelling request.

Confusing *ignore* with a spelling request can be quite damaging, since the system *must* initiate a spelling request to acquire an unknown word. The rightmost columns of the tables show that the manual model made a greater percentage of such confusions (10.0/15.4=65%) than the learned model (4.6/10.7=43%). However, when a spelling request was the ideal action, the lower right corners of the tables show that the learned model predicted it correctly 47% (5.0/10.7) of the time versus 19% (2.9/15.4) for the manual model.

The Compliance Problem

Since modeling the conflict problem proved promising, the same techniques were applied to address the *compliance problem*. A system was configured which supported both the main recognizer and a spelling recognizer in parallel, whenever a spelling was requested. A decision then had to be made as to which recognizer to trust.

In previous work, it was found that users do not always provide a speak-and-spell utterance when requested by the system (Filisko & Seneff 2004). In fact, it was shown in simulated user experiments that noncompliance (i.e., repeating the city name rather than spelling it) is a good strategy for the user when the city in question is known by the recognizer. Hence this option should be available to users.

```
1. (:ngram_score <= -34.1935)
  and (:total_score <= 93.2063)
   => action = noncompliant (240.0/0.0)
2. (:total_score <= 34.5113)
   => action = noncompliant (30.0/0.0)
3.  => action = compliant (330.0/0.0)
```

Figure 3: The rule set learned by JRip to predict user compliance in response to a spelling request from the system. The numbers in parentheses represent *coverage/errors* in the training data.

It was also discovered that users are inclined to simply spell the word when a speak-and-spell request is made. In fact, it seems that a simple spelling request is more natural. Therefore, the speak-and-spell recognizer was replaced by a simple spelling recognizer for these experiments. It is intuitively expected that acoustic and language model scores from a spelling recognizer would be poor if the user were to speak a sentence instead of spelling a word. It was therefore decided to supply these scores from the spelling recognizer to a learning algorithm for predicting user compliance.

The simulated user generated both a valid spelling and a noncompliant utterance for each of the 298 known and 596 unknown cities used to train the conflict problem model. This process resulted in 1,788 training samples.

After each utterance, the dialogue manager recorded the feature vector as well as whether the user was compliant or noncompliant. These data were processed by JRip to obtain a model consisting of three rules, as shown in Figure 3, where the possible predictions were *compliant* and *noncompliant*. Only the top-hypothesis *ngram_score* and *total_score* (*acoustic_* plus *ngram_score*) features appeared in the rules.

The learned model was tested with the same set of 50 scenarios used to test the conflict problem models; the simulated user was compliant for exactly one-half of the dialogues. However, there were, as expected, more predictions required for the noncompliant user since, as the user persisted in being noncompliant, the system would continue to request more spellings. All 57 compliant predictions and all 162 noncompliant predictions were correctly classified, yielding 100% accuracy for this task. But it remained to be seen if these results would hold up for real user data, which would likely be much more variable.

Experiment with Real Users

In order to provide a more realistic test of the learned models, a telephone-based dialogue system was developed to collect data from real users. The user's goal was to get the system to understand a source city and state, a destination city and state, and a date, in a flight reservation domain. Upon visiting a specified webpage, each user was tasked with calling the system and asking about flights according to a unique scenario provided. To minimize bias, no example utterances were suggested.

Before the data collection began, a set of scenarios was generated by pairing each of 50 known cities with each of 50 unknown cities, taken from the simulation test scenarios,

	NumSucc	AvgTurns	AvgAttrAcc
Manual	31/40 (77.5%)	8.97	93.0%
Learned	34/40 (85.0%)	8.50	96.0%

Table 5: Results for the *Manual* and *Learned* models in number of successful dialogues (*NumSucc*), average turns per successful dialogue (*AvgTurns*), and average attribute accuracy (*AvgAttrAcc*).

to yield a total of 2,500 scenarios. The source city in each scenario was always known to the recognizer, and the destination city was always unknown. A date for each scenario was selected at random.

To compare the performances of the manual and learned models for the conflict problem, the system guided each user to perform two sequential dialogues. The manual model was employed in the first dialogue for half of the sessions, while the learned model was used first in the remaining half. Once the first dialogue had ended (i.e., when the system recognized that the user had said, "Start over", as instructed), the system transparently switched to use the other model for the second dialogue. The user was asked to solve the same scenario both times, in order to minimize variability.

Forty-eight sessions were recorded by 34 unique users. Each dialogue was transcribed and tagged for ideal actions and user compliance by the first author. Each session ideally consisted of two dialogues; however, if a user hung up before starting the second dialogue, only one dialogue was available. Five such sessions were excluded from the analysis. Three more sessions were removed since at least one of the component dialogues was outside two standard deviations from the mean number of turns per dialogue. This filtering left a total of 40 valid sessions and 30 unique users.

Results

Table 5 shows that the learned model realized a greater number of successful dialogues, a slightly lower average number of turns per successful dialogue, and a greater average attribute accuracy than the manual model.

The results of a more detailed analysis are shown in Tables 6 and 7, which provide confusion matrices for the actions taken when the manual and learned models, respectively, were utilized. Overall, the learned model substantially outperformed the manual model with an accuracy of 84.1% versus 52.1%.

As observed with the simulated dialogues, the best performance for both models was achieved in the case of the *ignore* action. When the predicted action was *ignore*, the manual model was correct almost 66% (25/38) of the time, while the learned model showed over 86% (19/22) accuracy. The same argument applies in the case of real users as for simulated users: although *ignore* is the default action for the learned rules, both models predicted this action in roughly 50% of their total responses to the conflict problem.

Implicit and explicit confirmation were never predicted by the learned model. However, the manual model never predicted a truly ideal explicit confirmation, consistently confusing it with a spelling request. It also made many more confusions when *spelling request* was the ideal action. Such confusions can lower performance, since the relevant city

Reference→ *Prediction*↓	Ignore (52.1%)	Implicit (12.7%)	Explicit (5.6%)	Spelling (29.6%)
Ignore (53.5%)	25	4	2	7
Implicit (19.7%)	7	5	0	2
Explicit (7.1%)	0	0	0	5
Spelling (19.7%)	5	0	2	7

Table 6: Confusion matrix for the decision made using the manual decision model. An accuracy of 52.1% was obtained.

Reference→ *Prediction*↓	Ignore (52.3%)	Implicit (0.0%)	Explicit (0.0%)	Spelling (47.7%)
Ignore (50.0%)	19	0	0	3
Spelling (50.0%)	4	0	0	18

Table 7: Confusion matrix for the decision made using the learned decision model. An accuracy of 84.1% was obtained.

was unknown to the recognizer and spelling would be the only way to acquire the intended value. Any confusions would only lead to superfluous turns until a spelling request was finally initiated. While the manual model misclassified 67% (14/21) of the actions that should have been a spelling request, the learned model only misclassified 14% (3/21).

The percentage of *spelling request* predictions increased compared to the simulated dialogues from 6.4% to 19.7% for the manual model, and from 14.1% to 50.0% for the learned model, indicating the learned model is much more aggressive than the manual model in this respect. One possibility is that these large differences are due to an unrealistic or inadequately modeled simulated user.

A difference was also observed in the models' accuracies when a spelling request was predicted. In the simulated dialogues, the manual was more accurate than the learned model. However, with the real user dialogues, the learned and manual models were correct 82% (18/22) and 50% (7/14) of the time, respectively. It was common for both models to predict a spelling request when *ignore* was ideal; however, this type of confusion is much less damaging than others. If the system already believed the correct value, a spelling request would merely allow the user to clearly, albeit excessively, convey the intended city to the system.

A final noteworthy confusion is the manual model's tendency to predict *implicit confirm* when *ignore* is the ideal action. This confusion is certainly detrimental since the system consequently believes a spuriously hypothesized value, placing a burden on the user to initiate correction of the error. Users are often uncertain how to phrase a correction that will be understood by the system. Ameliorating such an error can further be complicated by the fact that the system believes the incorrect value with relatively high confidence. This situation could lead to frustrated users who give up before completing their tasks.

Assessment of the Compliance Problem

The compliance prediction model was deployed in the data collection system to assess its performance with real users. The results were very positive, showing an overall accuracy of 96.3%. For a total of 174 spelling requests, 77.6% belonged to compliant users who provided just the spelling of the city name as requested. Users were successful at spelling names ranging from *Rush* to *Ski Sundown Ski Area*.

Partially compliant user responses (e.g., "Stockbridge S T O C K B R I D G E") included a spelling among other words, and comprised 5.7% of the total responses. When such responses occurred, the recognizer hypothesized spurious letters for the words, but typically correct letters for the spelling. Thus, the system was usually able to determine the intended city following a spelling correction step.

A noncompliant user is one who does not provide a spelling as requested, but instead, for example, repeats the city name (e.g., "I wanna go to Yamhill, Oregon") or refuses to spell (e.g., "I don't wanna spell the name"). Nearly 17% of all responses were from noncompliant users.

Fifteen responses contained spellings (e.g., "The city is W A I M E A Waimea") but were uttered when a spelling was not requested. These responses are very interesting, since their format and timing are frequently unanticipated. A lesson here is that, once the system reveals to the user that it *can* accept spellings, it should be prepared for a spelling at *any* point in the dialogue.

Notable confusion occurred when a partial city spelling by the user (due to a clipped utterance) matched a city name in the database. One example of this is when the user was trying to spell *Myrtle Point*, but was cut off after spelling *Myrtle*. The system asked, "Did you say from Myrtle?", and the user replied, "Yes". In a similar example, the user's intended city was *Redwood Valley*, and when asked, "Did you say from Redwood?", the user simply continued spelling: "Uh V A L L E Y". Once discovered, such unanticipated responses can be handled in subsequent versions of a system.

Summary and Future Work

This paper presented an effective technique for learning strategies for resolving specific issues that arise in spoken dialogue systems, namely the *conflict problem* and the *compliance problem*. A simulated user was effective in providing training data for a RIPPER-based learning algorithm, and significant improvements were demonstrated over a baseline system that had used manually developed rules. Experimental results indicate that the learned model attained a much greater accuracy than the manual model in selecting the "ideal" action in response to the conflict problem. The learned model also made fewer detrimental confusions, contributing to its increased overall performance.

An additional assessment was performed on a model, learned from simulations, which predicted the compliance of a user in response to a system spelling request. The model performed very well, confirming its utility for this task.

In general, the results obtained for the dialogues with real users are comparable to those obtained with the simulated user. In some cases, better performance was observed for the real user dialogues, which is possibly a consequence of an inadequate simulated user model. More realistic behavior on the part of the simulated user could produce results more similar to those obtained with the real users. Such a possibility could be explored in future work.

The real user responses obtained in the data collection are valuable, not only for assessing the utility of the learned decision models, but also for providing real examples of user behavior when errors are encountered in an attribute acquisition dialogue. For example, the behavior in which a user spelled a multi-word city name across two utterances could be further explored via user simulation, and a strategy to handle such behavior could be developed.

Future work may also include applying this framework to learn decision models for other problems, such as predicting whether a user has spoken Mandarin or English in a multilingual dialogue system.

Acknowledgments

This research is sponsored by the T-Party Project, a joint research program between MIT and Quanta Computer Inc., Taiwan, and by Nokia.

References

Bohus, D., and Rudnicky, A. I. 2005. A Principled Approach for Rejection Threshold Optimization in Spoken Dialog Systems. In *Proc. SIGdial Workshop*, 2781–2784.

Cohen, W. W. 1995. Fast Effective Rule Induction. In *Proc. 12th Intl. Conference on Machine Learning*, 115–123.

Filisko, E., and Seneff, S. 2004. Error Detection and Recovery in Spoken Dialogue Systems. In *Proc. HLT/NAACL Workshop: Spoken Language Understanding for Conversational Systems*, 31–38.

Filisko, E., and Seneff, S. 2005. Developing City Name Acquisition Strategies in Spoken Dialogue Systems Via User Simulation. In *Proc. 6th SIGdial Workshop*, 144–155.

Henderson, J.; Lemon, O.; and Georgila, K. 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR Data. In *Proc. 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 68–75.

Higashinaka, R.; Sudoh, K.; and Nakano, M. 2005. Incorporating Discourse Features into Confidence Scoring of Intention Recognition Results in Spoken Dialogue Systems. In *Proc. ICASSP*, 25–28.

Hirschberg, J.; Litman, D.; and Swerts, M. 2001. Identifying User Corrections Automatically in Spoken Dialogue Systems. In *Proc. ACL*, 1–8.

Levin, E.; Pieraccini, R.; and Eckert, W. 2000. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. *IEEE Transactions on Speech and Audio Processing* 8(1):11–23.

Litman, D., and Pan, S. 2000. Predicting and Adapting to Poor Speech Recognition in a Spoken Dialogue System. In *Proc. AAAI*, 722–728.

Singh, S.; Litman, D.; Kearns, M.; and Walker, M. 2002. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research* 16:105–133.

van den Bosch, A.; Krahmer, E.; and Swerts, M. 2001. Detecting Problematic Turns in Human-Machine Interactions: Rule-Induction Versus Memory-Based Learning Approaches. In *Proc. ACL*.

Walker, M. A.; Langkilde-Geary, I.; Hastie, H. W.; Wright, J.; and Gorin, A. 2002. Automatically Training a Problematic Dialogue Predictor for a Spoken Dialogue System. *Journal of Artificial Intelligence Research* 16:293–319.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann, 2nd edition.