CONSTRUCTING ACCURATE BELIEFS IN SPOKEN DIALOG SYSTEMS

Dan Bohus *Alexander I. Rudnicky*

Computer Science Department,
Carnegie Mellon University,
Pittsburgh, PA, 15217

ABSTRACT

We propose a novel approach for constructing more accurate beliefs over concept values in spoken dialog systems by integrating information across multiple turns in the conversation. In particular, we focus our attention on updating the confidence score of the top hypothesis for a concept, in light of subsequent user responses to system confirmation actions. Our data-driven approach bridges previous work in confidence annotation and correction detection, providing a unified framework for belief updating. The approach significantly outperforms heuristic rules currently used in most spoken dialog systems.

1. INTRODUCTION

A key limitation in today's spoken language interfaces is their lack of robustness when faced with understanding errors. The large majority of such errors arise from the speech recognition process. The inherent difficulties of automatic speech recognition are further increased by the conditions under which spoken dialog systems typically operate: spontaneous speech, increasingly larger vocabularies, large user populations, variability in the quality of the input lines, etc. In these circumstances, average word error rates of 20% (and up to 40-50% for non-natives) are quite common. Unless mediated by better error awareness and robust recovery mechanisms, these errors propagate to subsequent stages of processing in a dialog system and exert a considerable negative impact on the quality and ultimately the success of the interaction [11,13].

Left unchecked, speech recognition errors can lead to two types of problems in a spoken dialog system: *non-understandings*, and *misunderstandings*. In a *non-understanding*, the system fails to obtain any semantic interpretation of the user's input. For instance, if the parsing stage lacks a certain degree of robustness, even the smallest speech recognition error can compromise the entire language understanding process. In contrast, in a *misunderstanding* the system does construct a semantic interpretation of the user's turn, but this interpretation is incorrect. In the absence of accurate mechanisms for detecting such errors, systems will take misunderstandings as fact and act based on invalid information.

We argue that, as a prerequisite for increasing robustness and making better decisions, spoken dialog systems must be able to accurately assess the reliability of the information they use. The work we present in this paper aims to endow these systems with the ability to ***construct more accurate beliefs by integrating information across multiple turns in the dialog***. Typically, spoken dialog systems rely on recognition confidence scores to guard against potential misunderstandings. While confidence scores can provide an ***initial*** assessment for the reliability of the information obtained from the user, ideally a system should leverage information available in subsequent user responses in order to ***update*** and ***improve the accuracy*** of its beliefs. Take for instance example 3 in Figure 1. Here, the system uses the recognition confidence score to form an initial belief about the start_time concept: the system believes that the start time is 3 p.m. with probability 0.65. Next, the system performs an implicit confirmation followed by a request for the date concept. The next recognition result is `"CAN YOU DETAILS TIME"`, which amounts to a non-understanding. In light of this interaction (as well as the initial confidence score) what should the system now believe about the start_time concept?

In this paper, we present a data-driven solution to this *belief updating problem*. We start by discussing related work in

Example 1: Explicit confirmation with correct value

1 S: Starting at what time do you need the room?
2 U: `[STARTING AT TEN A_M / 0.45]`
Starting at ten a.m.
3 S: **Did you say you wanted the room starting at ten a.m.?**
4 U: `[GUEST UNTIL ONE / 0.89]`
Yes until noon

Example 2: Implicit confirmation with correct value

1 S: For when do you need the room?
2 U: `[NEXT THURSDAY / 0.75]`
Next Thursday
3 S: **a room for Thursday, August 26th**... Starting at what time do you need the room?
4 U: `[FIVE TO SEVEN P_M / 0.58]`
Five to seven p.m.

Example 3: Implicit confirmation with incorrect value

1 S: How may I help you?
2 U: `[THREE TO RESERVE A ROOM / 0.65]`
I'd like to reserve a room
3 S: **starting at three p.m.** ... For which day do you need the conference room?
4 U: `[CAN YOU DETAILS TIME] - NONU/0.00`
I need a different time

Figure 1. Sample explicit and implicit confirmations. System turns are prefixed by (S); user turns are prefixed by (U); the speech recognition results and associated confidence scores are represented `[in between brackets]`

Section 2, and we argue that the heuristic approaches currently used for this purpose are suboptimal. Next, in Section 3 we precisely define and formalize the belief updating problem, and we introduce a reduced (yet useful in practice) version of the problem which we will address in the rest of the paper. In Section 4 we describe the data that we collected and used in this work, and in Section 5 we present an analysis of user responses to system confirmation actions. In Section 6 we present a simple machine learning approach for the belief updating problem. Experimental results indicate that the proposed approach constructs significantly more accurate beliefs than the heuristic rules currently used in most spoken dialog systems. Finally, in Section 7 we conclude and discuss future plans.

2. RELATED WORK

We have already noted *confidence annotation* as an important starting point towards continuous belief assessment in spoken dialog systems. Traditionally, confidence annotation schemes were focused on detecting word-level errors [1,4] and operated exclusively with information from the decoder (e.g. acoustic and language model and lattice information). More recently, various machine learning approaches have been proposed for detecting semantic errors (e.g. misunderstandings) and building semantic confidence scores [3,10] that are more appropriate for use in spoken dialog systems. These approaches use labeled in-domain data and typically integrate information from various sources of knowledge in the spoken dialog system. Generally, these techniques can be used to construct fairly accurate, but not perfect semantic confidence annotators.

Spoken dialog systems use the confidence score to form an initial assessment of the reliability of the concepts acquired from an input. Based on this score, a system can decide to accept the input, reject it, or engage in a confirmation strategy. Two confirmation strategies are widely used: *explicit confirmation* and *implicit confirmation* (see Figure 1). The tradeoff between these strategies is fairly well understood: explicit confirmations take an extra dialog turn; user responses to these confirmations are generally predictable and easy to handle. In contrast, implicit confirmations are more efficient if the value confirmed is indeed correct – no dialog turn is lost. However, an implicit confirmation with an incorrect value can increase confusion and create a greater cognitive load for the user [14]. This in turn can lead to a wider spectrum of possible user responses that can be more difficult to anticipate and therefore recognize correctly. Given these tradeoffs, most systems use implicit confirmations when they are fairly confident that the information to be verified is indeed correct and explicit confirmations when the confidence is lower. Additionally a system might decide to consider a concept grounded without taking any further action if the confidence score is very high, or reject the value altogether if the confidence score is very low.

Typically, very simple heuristic rules are used to update beliefs in light of user responses to various confirmation actions. For instance, after explicit confirmations, most systems consider the value grounded if they hear a ***yes***-type answer (e.g. *yes, correct, that's right*) or erase the hypothesis if they hear a ***no***-type answer (e.g. *no, that's wrong*). Additionally, if the user provides a new value for the concept, this will overwrite the old value. In the case of implicit confirmations, most systems rely on the user to overwrite the concept if the confirmed value is incorrect. We believe these heuristic approaches are suboptimal for a number of reasons. As example 3 in Figure 1 illustrates, users don't always overwrite slots. Previous studies have revealed that user responses following implicit confirmations cover a wide language spectrum [6], and simple heuristic rules will fall short on that account. Furthermore, user responses to confirmation actions are also subject to speech recognition errors, which renders the problem even more difficult for simple rule-based approaches (see for instance turn 4 from example 1 and example 3 in Figure 1). Finally, these heuristic rules lead to rather polarized beliefs (e.g. trust that value / don't trust that value), which do not accurately capture the degree of uncertainty the system should have. In light of the shallow solutions currently used for belief updating, results such as "users discovering errors through implicit confirmations are less likely to get back on track" [12] are not so surprising. The problem might not lie with the implicit confirmations per se, but rather with an inability to handle user responses and to accurately update beliefs.

Apart from *confidence annotation*, another vein of research that is relevant for this problem is *correction detection* [5,6]. Here, the task is to detect whether or not the user is currently attempting to correct a previous system misunderstanding. Machine learning techniques similar to the ones used for detecting misunderstandings are also used for detecting corrections. The resulting correction detectors are fairly accurate, but not perfect. Some similarities with the work we report in this paper can be found in previous work by Krahmer and Swerts [6]. In an effort to better understand the spectrum of user responses to system confirmation actions, Krahmer and Swerts analyzed positive and negative cues to implicit and explicit confirmations in a corpus of dialogs collected from a Dutch train-table system (we have repeated their analysis on data collected in a room-reservation system and we report the results in Section 5). Furthermore, Krahmer and Swerts showed that a memory-based learning approach can predict with a fairly high accuracy whether or not the previous confirmation contained an error. However, as the authors pointed out, their results were obtained on a small dataset, and using transcript information (as opposed to actual recognition results) as well as hand-annotated features, some of which are not available at run-time. Furthermore, Krahmer and Swerts were interested in detecting errors (a binary task), while we are interested in constructing accurate concept-level beliefs, which explicitly represent the degree of uncertainty the system has.

While both semantic confidence annotation and correction detection schemes are clearly valuable tools for belief updating, taken in isolation they do not provide a solution to this problem. We believe the solution lies in integrating confidence annotation and correction detection into a unified framework that would allow spoken dialog systems to continuously track their beliefs.

The idea of explicitly representing uncertainty and updating beliefs through time also appears in previous work by Paek and Horvitz [8]. In the DeepListener project, the authors used a Dynamic Bayesian Network to continuously update the belief over a user's goal in a simple command-and-control application. In contrast, in our work we are interested in updating beliefs over concept values in the context of a more complex task-oriented spoken dialog system. Furthermore, if in their work the network structure and parameters were handcrafted by domain experts, in our approach the model parameters are learned from data.

3. THE BELIEF UPDATING PROBLEM

We will now formalize the *belief updating problem*. Let us start by defining a few terms. Let ***C*** denote a *concept* that the system acquires from the user (e.g. date, start_time, end_time). By ***Belief$_t$(C)***, or *system belief over a concept* ***C***, we denote a representation of the system's uncertainty in the value of concept ***C*** at a certain time ***t***. Let ***SA(C)*** denote the system action at time ***t***, with respect to concept ***C***. For instance, in turn 3, example 2, Figure 1 the system implicitly confirms the date and requests the start time. Therefore, with respect to the date concept, the system action can be described as:

SA(date) = ImplicitConfirm(date) + Request(Other)

With respect to the start_time concept, we have:

SA(start_time) = ImplicitConfirm(Other) + Request(start_time)

Furthermore, let ***R*** denote the user response to the system action, as this response is perceived by the system. Then, the *belief updating problem* can be stated as follows:

given an initial belief over a concept ***Belief$_t$(C)***, *a system action* ***SA(C)*** *and a user response* ***R***, *compute the updated belief* ***Belief$_{t+1}$(C)***.

The belief over a concept is most accurately represented via a full probability distribution over all the possible concept values. However, this full-blown representation can be difficult to handle. From a practical perspective, it is very improbable that a spoken dialog system will "hear" throughout an interaction more than 3 or 4 conflicting values for a concept. For instance, in the corpus we collected (see Section 3), the maximum number of conflicting hypotheses accumulated through interaction for a concept was 3. Moreover, the system heard more than 1 hypothesis per concept in only 6.9% of cases. We believe that, even if a system used information from multiple recognition hypotheses (our system, like most other systems, did not), the number of different hypotheses that could be accumulated through interaction would rarely exceed 3 or 4.

Under these circumstances, a compressed representation of belief has the potential of greatly simplifying the problem without causing a corresponding decrement in performance. In this paper we report on the development of a belief updating scheme in which the belief over a concept is represented by the confidence score of the current top hypothesis for that concept. Moreover, we focus our attention only on updating beliefs in light of implicit or explicit confirmations (in the more general case the system could update its beliefs continuously, in all concepts, at all points in time). The reduced version of the belief updating problem that we use can therefore be stated as follows:

given an initial confidence score for the top hypothesis ***h*** *for a concept* ***C***, *construct an updated confidence score for the hypothesis* ***h***, *in light of the system confirmation action* ***SA(C)***, *and the follow-up user response* ***R***.

4. DATA

The work presented in this paper relies on data collected in an experiment [2] with RoomLine [9], a mixed-initiative spoken dialog system that assists users in making conference room reservations. The system finds the list of available rooms that satisfy an initial set of user-specified constraints (e.g. date and time, location, size, equipment, etc). The system then engages in a follow-up negotiation dialog to present this information to the user and identify which room best matches the user's needs. Sample interactions with the system are available online [9]. During the experiment 46 participants (first-time users) interacted with the system; each participant attempted a maximum of 10 scenario-based interactions. The scenarios had different degrees of difficulty and covered all aspects of the system's functionality. The collected corpus contains 449 dialogs and 8278 user turns.

The user speech was orthographically transcribed. Based on the transcriptions, we manually labeled the correct value for each concept (as specified by the user) at each turn, as well as whether or not the user was trying to perform a correction on each of the concepts present in the previous system turn.

To guard against potential misunderstandings, the system used both explicit and implicit confirmations. The strategies were engaged following a commonly used confidence threshold model (see Section 2). After the confirmation actions, the system used a set of simple heuristic rules to update the confidence score of the confirmed hypothesis. For explicit confirmations the system boosted the confidence score to 0.95 when the perceived user response contained a positive marker (e.g. *yes*, *right*) and deleted the hypothesis if the response included a negative marker (e.g. *no*, *wrong*) For implicit confirmations, the system would discredit the hypothesis if it heard a negative marker in the user response and the implicit confirmation was not followed by a yes/no question; if the user response included a new value for the confirmed concept, the system overwrote the old value (hence deleted the initial hypothesis); finally, for all other responses the system increased the score of the initial hypothesis to 0.95. Note that sometimes multiple confirmation actions were executed in a single turn (e.g. an implicit confirmation followed by an explicit confirmation on a different concept).

Apart from the explicit and implicit confirmations which where engaged by the error handling module, there were a number of other system turns that included information previously acquired from the user. For instance, the system might respond: "I found 10 rooms **this Friday** between **2** and **4 p.m.** Would you like a small room or a large one?" While the main purpose of this turn is to provide information to the user, the turn also acts as an implicit confirmation for the date, start_time and end_time concepts. In the sequel, we will refer to these as *unplanned implicit confirmations*. The system did not intend them as error handling actions; rather the implicit confirmations appeared as a side-effect of the design of system prompts. Moreover, the system did not update its belief over the confirmed concepts in this case. For these reasons, we analyzed these unplanned implicit confirmations separately from the planned ones.

5. USER RESPONSE ANALYSIS

To gain a better understanding of the confirmation strategies and the challenges that we are facing with respect to belief updating, we performed an analysis of user responses to these strategies.

Following Krahmer and Swerts [6], we divided user responses into three response-types according to whether they contain positive markers – ***yes*** (e.g. *yes, correct, right*), negative markers – ***no*** (e.g. *no, wrong*), or no positive or negative markers – ***other***. The classification was performed twice: once using the transcripts, and once using the output from the recognition system. We then cross-tabulated these classes against

A. Explicit Confirmation (from transcripts)			
	Yes	**No**	**Other**
Correct (1159)	**94%** [93 %]	**0%** [0%]	**5%** [7%]
Incorrect (279)	**1%** [6%]	**72%** [57%]	**27%** [37%]

C. Implicit Confirmation (from transcripts)			
	Yes	**No**	**Other**
Correct (554)	**30%** [0%]	**7%** [0%]	**63%** [100%]
Incorrect (229)	**6%** [0%]	**33%** [15%]	**61%** [85%]

E. Unplanned Implicit Conf. (transcripts)			
	Yes	**No**	**Other**
Correct (2725)	**25%**	**19%**	**56%**
Incorrect (457)	**12%**	**28%**	**60%**

B. Explicit Confirmation (from decoded results)			
	Yes	**No**	**Other**
Correct (1159)	**87%**	**1%**	**12%**
Incorrect (279)	**1%**	**61%**	**38%**

D. Implicit Confirmation (from decoded results)			
	Yes	**No**	**Other**
Correct (554)	**28%**	**5%**	**67%**
Incorrect (229)	**7%**	**27%**	**66%**

F. Unplanned Implicit Conf. (decoded)			
	Yes	**No**	**Other**
Correct (2725)	**24%**	**17%**	**59%**
Incorrect (457)	**10%**	**21%**	**69%**

Table 1. Cross-tabulation of response-type against correct and incorrect system confirmation actions. Percentage numbers reflect proportions from the correct and respectively incorrect confirmations. Bracketed numbers were reported by Krahmer and Swerts in a similar study of a Dutch train-timetable information system [6].

whether the system was confirming a correct or an incorrect concept value. The results are presented in Table 1.A-F.

As Table 1.A shows, user responses following explicit confirmations included positive markers in 94% of the cases when the valued confirmed was indeed correct. In contrast, the answers included a negative marker in only 72% of the explicit confirmations with incorrect values. This discrepancy is consistent with prior observations made by Krahmer and Swerts (presented in brackets in the Table 1.A). It indicates that in a significant number of cases (27%) users attempted to correct the system by other means than a negative answer to the explicit confirmation question. A closer look at these instances reveals that in most of these cases (~70%) users repeated the correct value for the concept. Other times, users ignored the system's explicit confirmation and tried to change the focus of the conversation. Prompted by this observation, we cross-tabulated users' correction attempts versus the correctness of the value the system tried to verify. The results are shown in Table 2.A. We noted that 10% of the incorrect explicit confirmations (29 of the 29+250=279 cases) were not corrected by the users – these are mostly the cases in which users were trying to shift the focus of the conversation. Finally there were a small number of *other*-type responses on explicit confirmations which stemmed from audio end-pointing errors and turn-overtaking issues.

A comparison between Tables 1.A and 1.B reveals an increased proportion of the *other*-type responses in the decoded results. This difference is explained by misrecognitions of YES / NO into acoustically similar words (e.g. THIS / SMALL). The difference is also present for both planned and unplanned implicit confirmations, and highlights again the fact that the presence of recognition errors increases the difficulties of detecting corrections and building accurate beliefs.

For implicit confirmations, there are much fewer *yes*- and *no*-type responses, and the *other*-type responses dominate (see Figure 1.C). For correct implicit confirmations this large number is expected – users mostly answer the follow-up system question.

A. Explicit Confirmation		
	User corrects	**User does not correct**
Correct	**0**	**1159**
Incorrect	**250**	**29**

B. Implicit Confirmation		
	User corrects	**User does not correct**
Correct	**2**	**552**
Incorrect	**111**	**118**

C. Unplanned Implicit Confirmation		
	User correct	**User does not correct**
Correct	**11**	**2714**
Incorrect	**138**	**319**

Table 2. Users' correction attempts versus correctness of confirmed values.

The large number of *other*-type responses on incorrect implicit confirmations is explained by the fact that oftentimes users do not attempt to correct the incorrect concept values implicitly confirmed by the system. Table 2.B. shows that for 51.5% (118 / 118+111=229) of incorrect implicit confirmations users are not attempting to correct the system.

A closer analysis of these instances led to several observations. First, although in these 118 cases users did not immediately correct the system's incorrect implicit confirmation, in 49 of these cases (~40%) users did attempt to correct the error in a later turn. More interestingly, the analysis revealed that users interact with the system strategically: they correct the errors which *have to be recovered* for the successful completion of the task (we shall refer to these as *critical errors*), and mostly ignore the others. Given the nature of the domain, certain scenarios could be completed by users in different ways. This in turn rendered some concepts more important for task success than others. For instance, if the system accidentally mis-recognized that the user wanted a room with a whiteboard, this incorrectly filled concept (equipment) had no impact on task success if the goal was only to make a reservation for a room on Friday morning from 10 to 11. A user could ignore this system error altogether and still complete the task successfully. Out of the 49 later-turn corrections, 47 were on critical errors, and only 2 on non-critical errors. At the same time, there were only 14 critical errors that were never corrected, and hence had a direct contribution to task failure. These results confirm that users adapt their behaviors to their specific goals, and engage in corrections only when it is necessary to do so.

Finally, for unplanned implicit confirmations, the distribution of response types is similar to the one for the planned implicit confirmations, and the same phenomenon of users not always correcting the system can be observed.

6. EXPERIMENTS AND RESULTS

6.1. Approach

The reduced form of the belief updating problem we address can be stated as follows: given an initial confidence score for a concept value to be confirmed, a system confirmation action, and the user response to that system action (as it was perceived by the system), compute an updated and more accurate confidence score for the value undergoing the confirmation. We used a machine learning approach.

While detection of misunderstandings and corrections can be regarded as a binary classification task, our goal is to generate good probability estimates (a continuous value between 0 and 1 which represents the system's uncertainty). The "goodness" of these probability estimates can be evaluated by computing the cross-entropy between the true distribution of the target values and the distribution predicted by the learned model (the smaller the cross-entropy, the better). When the target is binary, cross-entropy equals the negative average log-posterior of the correct class. This metric is commonly referred to as soft-error. Discriminant classification techniques traditionally used for detecting misunderstandings and corrections (e.g. decision trees, memory-based learning, etc.) are inappropriate in this case since they focus on minimizing hard- rather than soft-error.

A machine learning technique with good soft-metric performance is binary logistic regression. This method assumes a certain density model and learns parameters such as to maximize the average log-posterior of the correct class. Furthermore, this technique is sample efficient, and, when used in a stepwise approach, can also perform feature selection. We therefore decided to use a collection of stepwise logistic regression models. For each system action (explicit confirmation, implicit confirmation and unplanned implicit confirmation), we partitioned the data according to the response-type of the recognition result (e.g. *yes, no, other)* and then trained a separate stepwise logistic regression model for each partition. The rationale for training separate models was that features are likely to work differently for each response-type. The proposed approach is equivalent with constructing a one-level logistic model tree (LMT) [7] for each system action, with the root node splitting on the response-type.

6.2. Dataset: features and labels

Our dataset contained 5403 data-points (1438 for explicit confirmation, 783 for implicit confirmation and 3182 for unplanned implicit confirmation). Each data-point describes a *concept update*: it specifies the concept and value undergoing the update, the initial system belief about that concept, the system confirmation action with respect to that concept, and a label which indicates if the value undergoing the confirmation was indeed correct or not (this was our target for learning).

We identified a large number of features from different levels in the system, which could be relevant for the task at hand. All these features are available to the system at run-time. Given the space constraints, we only summarize the feature set below[1].

- **Initial situation features**: initial confidence score of the top hypothesis, the identity of the concept undergoing the update, current dialog state, turn number;
- **System action features:** features describing other system actions taken in conjunction with the current confirmation;
- **Acoustic and prosodic features of the user response:** duration, pitch (mean, min, max, range, min and max slope, plus normalized versions), speech rate, initial pause, etc.;
- **Lexical features of the user response:** number of words, the presence / absence of lexical terms highly correlated with corrections; the terms were apriori identified by computing mutual information between these terms and corrections;
- **Grammatical features of the user response:** number of grammar slots contained in the parse, the number of new slots, repeated slots, goodness of parse scores, etc.;
- **Dialog level features of the user response:** features describing how well the response matched the current system expectation, whether the response contained a new value for the concept, etc.

[1] For the complete feature set, see: www.cs.cmu.edu/~dbohus/blf_upd/

6.3. Results

We trained the logistic model trees and evaluated them using 10-fold cross validation. We compared the cross-validation results (both hard- and soft-error) against three baselines. The results are shown in Figure 1. The *initial* baseline reflects the accuracy of the initial system belief, before any update is performed. The *heuristic* baseline reflects the performance attained by the heuristic update rule currently used by the system (see Section 4). Finally, we also report an *oracle* baseline. The user-response analysis from Section 5 revealed that users do not always correct system errors. As a result, even if we knew precisely when the user is correcting the system, we would not be able to construct absolutely perfect beliefs. If for instance we assumed that the concept is incorrect each time the user makes a correction we would still make a certain number of errors. The *oracle* baseline reflects the performance we would attain if we knew precisely when the user is correcting the system.

As Figure 2 shows, the logistic model tree (LMT) produces significant improvements over the heuristic rule on both hard- and soft-error. For explicit confirmations, the simple heuristic rule used by the system bridges a large portion of the gap between the no-update and the oracle baseline: from 31.15% to 8.41%. However, the logistic model tree performs even better, further reducing the hard-error rate from 8.41% to 3.57% ($p=0.029$), very close to the oracle performance of 2.71%. An even larger improvement is attained for updates after implicit confirmations. As expected, the heuristic rule faces more

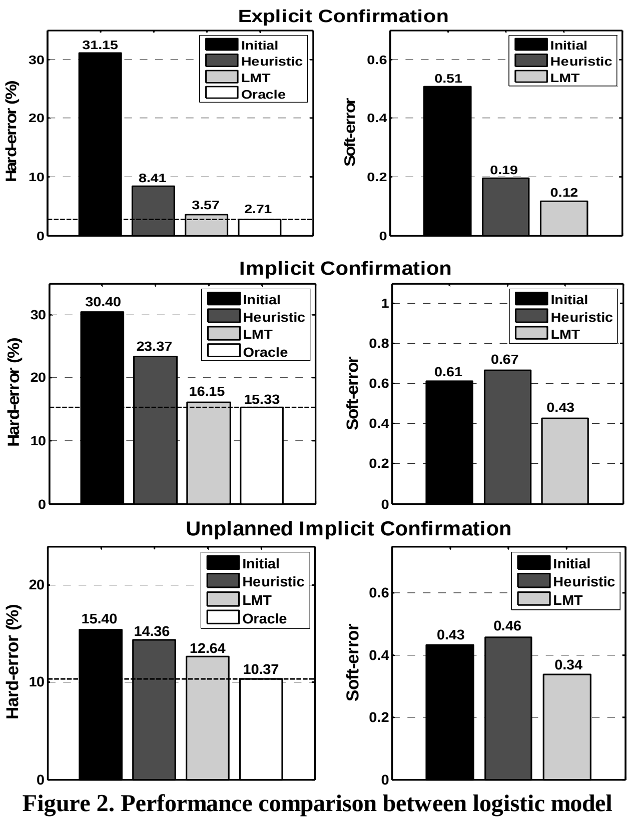

Figure 2. Performance comparison between logistic model tree (LMT) and several baselines (both hard and soft-error)

difficulties in this situation, and reduces the hard-error rate only from 30.4% to 23.37%. The logistic model tree again bridges the remaining gap from 23.37% to 16.15% (p=0.0094), yielding results very close to the oracle baseline (15.33%). For unplanned implicit confirmations, the belief updating problem is even more difficult. Usually in this case multiple concepts are confirmed in a single turn, and even if our features capture the fact that the user attempts to correct, it is often difficult to infer which concept is being corrected. The heuristic rule provides almost no improvement in performance over the initial baseline (the minor improvement comes from instances where the user is repeating the correct value and thus overwriting the concept). The learning approach produces a better result, although this time it bridges only half of the gap between the initial and oracle baselines, reducing the hard error from 15.4% to 12.64% (p=0.07).

The discussion above has focused on hard-error. However, as we argued in subsection 6.1 we are really interested in good soft-error performance. Indeed, the proposed learning approach brings statistically significant improvements on the soft metric for each of the three confirmation types (see plots on the right-hand side in Figure 2). It is interesting to notice that in the case of implicit confirmations (both planned and unplanned) the heuristic rule does not bring any gains; this happens largely due to the fact that the heuristically-updated beliefs are highly polarized (probabilities close to 0 or 1), and do not accurately reflect the uncertainty the system has.

Some of the most informative features for belief updating were: the initial confidence score, various prosodical features, expectation match, barge-in, various lexical features, the presence of repeated grammar slots, as well as the identity of the concept to be confirmed (concept-id). In fact, adding the concept-id led to significant gains in accuracy for belief updating following implicit confirmations. We believe this is related to our observations from Section 5. Since some of the concepts (e.g. date, start_time, end_time) are on average more important than others for the task at hand, adding knowledge into the belief updating process about the identity of the concept undergoing the confirmation helps the system better take into account the user responses to the implicit confirmation.

7. CONCLUSION

We have proposed a machine learning based approach for constructing more accurate beliefs in spoken dialog systems by integrating information over multiple turns in the conversation. In particular, we are interested in updating system beliefs in light of user responses to explicit and implicit confirmations. We have addressed a simplified yet practically very useful form of the belief updating problem, whereby the system belief is reduced to the confidence score of the top hypothesis. We have shown that a logistic model tree using features from different levels in the system can be trained to construct beliefs that are significantly more accurate than those produced by simple heuristic rules typically found in current systems. Our work has brought together previous insights from semantic confidence annotation and correction detection into a unified framework that allows spoken dialog systems to more accurately track their beliefs and avoid misunderstandings.

In the process, our data analysis has corroborated prior studies that have shown that user responses to explicit and implicit confirmations cover a wide language spectrum. Furthermore, we have found that oftentimes users do not correct implicit confirmation actions containing incorrect values, unless recovery is essential for the task at hand. Rather, users consider the overall properties of the task and adjust their strategies accordingly.

In light of the results we obtained on the reduced version of the belief updating problem, we are currently extending this work in several directions. First, we plan to use a more comprehensive, but still compact belief representation: n hypotheses plus "other", where n is a small number like 2 or 3 (the reduced version of the problem we addressed in this paper had n=1). Secondly, we plan to extract and use information from multiple hypotheses from the recognition engine. Finally, since users can attempt corrections at any point in the dialog, we plan to investigate the possibility of continuously updating beliefs, after all system actions (i.e. not only after confirmation actions).

8. REFERENCES

[1] D. Bansal, and M.K. Ravishankar, *"New Features for Confidence Annotation"*, in Proceedings of ICSLP-98

[2] D. Bohus, and A. Rudnicky, *"Sorry, I didn't Catch That: An Analysis of Non-understandings and Recovery Strategies"*, Proceedings of SIGdial-06, Lisbon, Portugal, 2006.

[3] P. Carpenter, C. Jin, D. Wilson, R. Zhang, D. Bohus and A. Rudnicky, *"Is This Conversation on Track?"* in Proceedings of Eurospeech-2001.

[4] S. Cox, and R. Rose, *"Confidence Measures for the Switchboard Database"*, in Proceedings of ICASSP-96

[5] J. Hirschberg, D. Litman, M. Swerts, *"Identifying User Corrections Automatically in Spoken Dialogue Systems"*, in Proceedings of NAACL'01, Pittsburgh, PA, June 2001

[6] E. Krahmer, M. Swerts, M. Theune, and M. Weegels, *"Error Detection in Spoken Human Machine Interaction"*, in International Journal of Speech Technology, Vol.4, No.1, 19-29.

[7] N. Landwehr, H. Mark, and F. Eibe, *"Logistic Model Trees"*, in Proceedings of ECML 2003, Croatia, 2003.

[8] T. Paek, and E. Horvitz, *"DeepListener: Harnessing expected utility to guide clarification dialog in spoken language systems"*, in Proceedings of ICSLP'2000, Beijing, China, 2000.

[9] RoomLine: www.cs.cmu.edu/~dbohus/RoomLine

[10] R. San-Segundo, B. Pellom, and W. Ward, *"Confidence Measures for Dialogue Management in the CU Communicator System"*, in Proceedings of ICASSP-2000.

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

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

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

[14] M. Weegels, *"User's Conceptions of Voice-Operated Information Services"*, in International Journal of Speech Technology, 3(2):75-82.