Error Awareness and Recovery in Task-Oriented Spoken Dialogue Systems

- Ph.D. Thesis Proposal -

Dan Bohus

Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, 15213

Abstract. A persistent and important problem in spoken language interfaces is their lack of robustness when faced with understanding errors. The problem is present across all domains and interaction types, and stems primarily from the unreliability of the speech recognition process. I propose to alleviate this problem by (1) endowing spoken dialogue systems with better error awareness, (2) constructing a richer repertoire of error recovery strategies, and (3) developing a practical data-driven approach for making error handling decisions. The proposed work will address questions and make contributions in each of these three areas. For the first part, I propose to develop a belief updating mechanism that integrates confidence annotation and correction detection into a unified framework, and allows spoken dialogue systems to continuously track the reliability of the information they use. For the second part, I propose to implement and investigate an extended set of error recovery strategies addressing common problems in human-computer dialogue. Finally, I plan to bring these two capabilities together in a scalable reinforcement-learning based approach for making error handling decisions in task-oriented spoken dialogue systems.

1. Introduction

Over the recent years, advances in automatic speech recognition, as well as language understanding, generation, and speech synthesis have paved the way for the emergence of complex, task-oriented spoken dialogue systems. Here are some examples: Jupiter (MIT) [76] provides up-to-date weather information over the telephone. CMU Communicator (CMU) [49] acts as a travel planning agent and can arrange multi-leg itineraries and make hotel and car reservations. TOOT (AT&T) [61] gives spoken access to train schedules. Presenter (Microsoft) [42] provides a continuous listening command and control interface to PowerPoint presentations. WITAS (Stanford) [32] provides a spoken language interface to an autonomous robot helicopter. AdApt (KTH) [22] provides real-estate information in the Stockholm area. TRIPS (Rochester) [19] is a spoken-language enabled planning assistant. I have named only a few; for a longer list, see [59].

Traditionally, the research community has focused on building spoken dialogue systems for information access and command-and-control tasks. Current initiatives aim at the development of more sophisticated language-enabled interactive agents, such as personal assistants (e.g. CALO [11], LARRI [6]), taskable agents (e.g. Orion [55]), interactive tutors (e.g. Paco [47]), call-routing systems (e.g. How May I Help You? [20]), embodied conversational agents (e.g. Rea [13], Olga [60]), etc. At the other end of the complexity spectrum, simpler systems have already transitioned into day-to-day use. United Airlines handles claims for missing and delayed luggage via a spoken dialogue system called Simon. Newsweek uses one for subscription renewal and cancellations. Various companies (e.g. Speechworks, TellMe, BeVocal, HeyAnita) are successfully commercializing platforms and applications to deliver voice dialing, messaging and a large variety of telephone-based automatic customer care services.

The development of these systems has benefited from but at the same time prompted numerous advances in the component technologies. Speech recognition accuracy has improved over time, allowing spoken dialogue systems to handle larger vocabularies and user populations. Dialogue management paradigms, such as finite-state, form-filling, task or plan-based dialogue have been proposed and empirically validated in the field. Novel text-to-speech synthesis techniques, such as unit-selection, have allowed the development of natural voices in limited domains. A certain consensus has been reached with respect to how to architecturally structure a spoken dialogue system. Nevertheless, a lot of problems remain in need of better solutions. Automatic recognition and understanding of spontaneous speech are still far from perfect and as a result spoken language interfaces remain very brittle. Current solutions for system development do not transfer well across tasks and domains, and require considerable amounts of human effort and expertise. More complex aspects of human-computer interaction, such as multi-modality, multi-party conversations, embodied interfaces, fine-granularity timing and turn-taking are just beginning to be addressed.

One of the most important problems in the development of spoken language interfaces is their lack of robustness when faced with understanding errors. The problem is present across all domains and interaction types, and stems from the inherent unreliability of the recognition process. The recognition problems are exacerbated by the conditions under which spoken language interfaces typically operate: spontaneous speech, increasingly larger vocabularies, and large variations in user populations and in the quality of the input lines. Speech recognition errors, unless mediated by a better awareness and robust recovery mechanisms, can propagate to the higher levels of these systems and exert a considerable impact on the quality and success of the interactions.

I will illustrate these problems with an example. The sample conversation that follows is an excerpt from an interaction with an early version of the CMU Communicator [49], a spoken dialogue system operating in the air-travel planning domain. The actual recognition results are shown in square brackets, immediately following the user's utterances.

```
1  System: Are you a registered user?
2  User:   No I'm not [NO I'M NOT NO]
3  System: What is your full name?
4  User:   Adam Sandler [ATHENS IN AKRON]
5  System: There is an Athens in Georgia and in Greece. Which
           destination did you want?
6  User:   Neither [NEITHER]
7  System: I'm sorry, I didn't catch that. There is an Athens in
```

Georgia and in Greece. Which destination did you want?

8 User: *Georgia* [GEORGIA]
9 System: A flight from Athens… Where do you want to go?
10 User: *Start over* [START OVER]
11 System: Do you really want to start over?
12 User: *Yes* [YES]
13 System: What city are you leaving from?
14 User: *Hamilton, Ontario* [HILTON ONTARIO]
15 System: Sorry, I'm not sure I understood what you said. Where are you leaving from?
16 User: *Hamilton* [HILTON]
17 System: I'm still having trouble understanding you. To go on, I need you to answer the following question. What city are you leaving from?
18 User: *Toronto* [TORONTO]

Let's analyze this example in more detail. In turn 3, the system asks for the user's name. The user responds in turn 4. The name is not in the system's vocabulary, and the speech recognition engine produces an erroneous result: [ATHENS IN AKRON]. Without the ability to assess the reliability of its inputs, or to make better use of current expectations in dialogue, the system takes this misrecognition as fact; a ***misunderstanding*** has occurred. Since the dialogue manager knows of two cities by the name of Athens, one in Greece and one in Georgia, it starts a sub-dialogue (turns 5 through 9) to establish which city the user is referring to. Given that the user never said anything about Athens, Georgia or Greece, the system's question in turn 5 probably appears confusing and nonsensical. The system's behavior is opaque, and lacks any explanation of state or choice of actions (e.g. "I thought you said Athens"). In turn 6 the user responds to the disambiguation question, rejecting both proposed options. Although the recognition and language understanding components produce the correct result this time – [NEITHER], the dialogue management layer is not equipped to handle this third option and rejects the user's utterance; in this case, a ***non-understanding*** has occurred.

By turn 10, the user decides to restart the conversation by using the *Start Over* command. The dialogue is restarted, but in turn 14 the system misrecognizes *Hamilton, Ontario* for [HILTON ONTARIO]. Hamilton is another out-of-vocabulary word. Luckily, at this early point in the interaction the system is not prepared to accept information about hotel preferences; the dialogue manager makes judicious use of expectations and this last misrecognition is not acted upon. Rather, the system regards the input as another non-understanding, rejects the utterance, and, for lack of any better strategy, repeats the question (turn 15). The user repeats the city name. The same recognition error appears again, and the system repeats its question one more time. By turn 18 the user probably realizes that the system is not likely to understand *Hamilton*, and adapts to the situation by changing his intended departure place.

Unfortunately, this is not just a contrived corner-case interaction carefully chosen to make a point. The example was indeed selected to illustrate a large spectrum of problems over a short number of turns, but this type of breakdowns are not uncommon at all in any state-of-the-art spoken dialogue system.

Here are some statistics. An analysis of the CMU Communicator corpus indicated that 32% of the utterances contain understanding errors [12]. Similar results are reported in the literature for other spoken dialogue systems. Research on confidence annotation shows that typically 20 - 40% of the utterances addressed to a system are not understood correctly (see Table 1). Other work shows that approximately a quarter to one third of the utterances addressed to a system are dedicated to correcting system mistakes: 32% in a Dutch spoken dialogue system that provides train time-table information [31]; 29% in TOOT [61]. Moreover, due to changes in speaking style, corrections are themselves two to three times more likely to be misrecognized that non-corrections [34, 61].

The relatively large numbers of understanding errors and corresponding user corrections have a significant impact on the interaction as a whole. In the CMU Communicator, the 32% understanding error rate results in 66% of the sessions having at least one understanding error. In 40% of these cases (26% of the total number of sessions), the end result was complete breakdown in interaction and failure to achieve the task. The users managed to get the interaction back on track in only 60% of the sessions containing misunderstandings. In a similar error analysis study, Shin and Narayanan [56] found 235 error segments in 141 dialogues in the air-travel planning domain; this equates to an average of 1.66 error segments per session. 22% of the error segments (37% of the sessions) led to a complete breakdown of the interaction.

System	Semantic error rate
CMU Communicator [12]	32%
CU Communicator [50]	27%
How May I Help You [71]	36%
Jupiter [23]	28%
SpeechActs [74]	25%

Table 1. Proportion of utterances containing understanding errors in various spoken dialogue systems

The statistics discussed above, as well as a large amount of anecdotal evidence show that these problems occur in significant numbers across most domains and interaction types. Unless properly addressed, they have a major negative impact on the usability and performance of spoken dialogue systems.

The research program proposed in this document aims to alleviate these problems by endowing spoken dialogue systems with improved error awareness and a richer repertoire of error handling strategies, and by developing a task-independent, adaptive and scalable framework for making error handling decisions.

The rest of the document is organized as follows: in the next section, I discuss in more detail the major factors and current limitations leading to interaction breakdowns in spoken dialogue systems. Section 3 compares two possible approaches for increasing robustness, leading in turn to an outline of the proposed research program, presented in Section 4. Section 5 reviews the relevant work published to date in the literature. Section 6 presents each of the proposed work items in detail. Finally, Section 7 concludes the document with the anticipated timeline for the proposed research program.

2. Factors and Limitations Underlying Interaction Breakdowns

The speech recognition process is the primary source of errors in spoken dialogue systems. Would speech recognizers produce perfect results, the large majority of the problems would disappear[1]. We would mostly be left with the task of managing the dialogue under assumed perfect inputs, a problem for which various approaches have been proposed and are evolving in the community. In reality, automatic recognition of spontaneous speech is still imperfect at best. In the example discussed, the recognition errors were triggered by the use of out-of-vocabulary words; this is however just one of a large variety of factors that can trigger such errors. Small changes in the environment, microphone or telephone line quality can seriously impair recognition performance. Since spoken dialogue systems are generally targeted to large user populations, speaker variability represents another major concern. Adding to this mix the disfluencies which characterize spoken language, such as stutters, false-starts, restarts, coughs, lip-smacks and various other non-lexical acoustic events, the recognition error rates can jump from a typical 8-10% for read speech to as high as 30% for spontaneous speech.

Recognition accuracy has a large impact on the performance of spoken dialogue systems [51, 69, 70]. After task completion, recognition accuracy was repeatedly shown to be the most important predictor of user satisfaction [69, 70]. Other studies involving the tradeoff between recognition accuracy and system flexibility have shown that users currently prefer less flexible, more constrained (i.e. system-initiative), but more accurate systems [68]. The impact of recognition errors is so pronounced because systems are not well prepared to handle the resulting uncertainties. Two important limitations which can be easily identified are shortcomings in the ability to accurately detect and diagnose problems, and a lack of sufficient and effective error handling strategies. Moreover, a principled yet practical computational framework for making decisions for engaging in error handling strategies is still missing. In the absence of these competencies, the inherent unreliability of the speech recognition process exerts a strong negative influence on the performance of spoken dialogue systems, and severely limits the naturalness of the interaction and the complexity of the tasks that can be addressed. In the reminder of this section, I will discuss each of these deficiencies in more detail.

A first important deficiency which can be observed in many current spoken dialogue systems is their inability to accurately identify and diagnose problems. Left unchecked, speech recognition errors can generate to two types of problems: *non-understandings* and *misunderstandings*. A *non-understanding* occurs when the system fails to acquire any useful information from the user's turn. For instance, if the parsing layer lacks a certain degree of robustness, then even the smallest recognition error can ruin the whole language understanding process. In such cases no useful semantic representation of the input can be obtained, and we have a non-understanding. Non-understandings can also occur at the pragmatic level when the dialogue manager cannot integrate the information obtained from the user's turn into the current context. By contrast, a *misunderstanding* occurs when the system incorporates information from the user's turn, but that information is incorrect. This generally happens when the language understanding layer generates a plausible semantic representation, which happens to match the context, but which does not correspond to the user's intention.

[1] Other sources of understanding errors exist besides the recognition process. Parsing (i.e. grammar coverage) errors can lead to misunderstandings or non-understandings as well. High-level interpretation, reference resolution, and intention recognition can also create similar problems. Nevertheless, in most spoken dialogue systems, the largest proportion of errors stem from the speech recognition process [8, 12].

In the absence of a mechanism for assessing the reliability of their inputs, systems will take misunderstandings as fact and will act based on invalid information. We have seen this happening in turn 4 of the example presented in Section 1: the system believed that the user had said [ATHENS], and incorrectly engaged in a sub-dialogue to handle that information. The system was unaware of the problem because it blindly trusted its inputs, unable to assess and make use of the reliability of the information it heard. Had the system been able to detect that the recognition result from turn 4 was not reliable, it might have avoided the off-track sub-dialogue in turns 5-12 (including the user initiated start-over).

When a non-understanding occurs, no false information is incorporated into the system, but the situation is not much better in this case either. Without a diagnosis process that can indicate the likely sources of the non-understanding, the system's follow-up options are limited and uninformed. In the given example they amounted to asking the question again and hoping for better luck (turns 15 and 17). A system that can detect the possible source of the error (was it an out-of-vocabulary word? a transient noise? does the user speak with a pronounced accent? does the user not know what to say? etc) could make more informed decisions, and therefore be more likely to succeed in setting the dialogue back on track. The ability to assess how well the interaction is proceeding within a given turn, a certain discourse segment, or over the whole conversation, as well as to diagnose potential problems at each of these levels would provide a better basis for making error handling decisions.

A second important limitation which leads to interaction breakdowns is the lack of effective conversational strategies for preventing and recovering from errors. When faced with misunderstandings or non-understandings, humans use a large variety of strategies to set a conversation back on track [58, 75]: various forms of explicit and implicit confirmation, requests for disambiguation, requests for repeating information, checking context, trying alternative plans to achieve the same dialogue goal, even guessing or ignoring unreliable information. More strategies can be envisioned in the context of a task-oriented human-machine conversation: lexically entraining the users, providing help, falling back to a stricter initiative and constraining the input language, relying on alternate input modalities such as DTMF (touch-tone) or pen-based input, etc. Only a few of these strategies have been thoroughly investigated and are consistently used in today's spoken dialogue systems.

The sample conversation from Section 1 illustrates the negative effects of this deficiency. The disambiguation strategy was poorly implemented: the system was not able to handle the [NEITHER] response from the user in turn 6. Similarly, the implementation of the implicit confirmation in turn 9 – "A flight from Athens…" – lacks the necessary precision. "A flight from Athens, Georgia…" would have been more informative, and less confusing to the user. Last but not least, each time the system encountered a non-understanding, the only available strategy – asking the user to repeat – was not conducive to reestablishing the conversational flow. Other error handling strategies, such as switching to DTMF, or changing to an alternative dialogue plan (e.g. asking the user for the state first and then providing the list of available airports) would probably have been more effective in this situation.

Thirdly, a principled, yet practical computation framework for making the error handling decisions is still missing. The very problem of error handling is often regarded as an add-on, and the large majority of systems use handcrafted heuristic decision rules to engage in a small number of error handling strategies. This approach lacks a solid basis, and cannot provide any optimality guarantees. Moreover, the technique cannot be easily extended to handle a larger number of strategies, since the tradeoffs that need to be solved become ever more complex. More systematic approaches, based on Bayesian decision theory and reinforcement learning,

have been proposed but so far they lack scalability are not applicable in large, practical spoken dialogue systems.

Finally, the factors and limitations discussed above are not the only contributors to the brittleness observed in today's spoken dialogue systems. Other factors, such as inconsistency and poor design of the system's actions at the task level, deficiencies in the user's knowledge about the system's functionality, lack of basic conversational skills on the system side (i.e. timing, turn-taking, handling negations, repeat, etc.), and lack of support for user-initiated repairs can also lead to interaction breakdowns.

3. Alternatives for Increasing Robustness

The discussion from the previous section highlights two different pathways for increasing the robustness of spoken dialogue systems. One approach is to increase the accuracy of the speech recognition process, in an effort to eliminate the errors altogether. A different approach is to assume that the inputs will always be unreliable, and create the mechanisms for acting robustly at the dialogue management level.

Reducing the speech recognition error rates would certainly lead to corresponding improvements in the performance of spoken language interfaces. Not surprisingly, the problem has received a great deal of attention. Numerous efforts are directed at improving the baseline speech recognition performance: more robust representations, more sophisticated statistical models, better search techniques, and better acoustic and language modeling techniques. The typical results so far have been consistent but small incremental improvements. In contrast, the most effective and widely applied technique for improving recognition accuracy in spoken dialogue systems has been remarkably unsophisticated – constrain the input language. By imposing limits on what the users can say at any given point in the conversation, the problem is simplified and the decoding process becomes less prone to errors. A widely used technique is to build systems which always keep the initiative, and constrain the user responses based on the current focus of the conversation. Other techniques, such as interpolating state-specific and generic language models [73], or backing off from grammar-based to SLM-based recognition, can achieve a more accurate control of the tradeoff and regain some of the lost flexibility while still improving recognition performance. The Universal Speech Interface [48, 63] is another approach, still aimed at alleviating the recognition problem. In this case the constraints are on the form rather than the contents of the user's language. Users are trained to speak in a stylized fashion, within the bounds of a well-defined universal grammar. In the end, all these techniques trade off naturalness and flexibility for accuracy. They are applicable in a number of relatively simple domains and interaction types, and can indeed lead to increased robustness. However, if the ultimate goal is to build natural, seamless spoken language interfaces to complex domains, these techniques do not provide a suitable solution.

The alternative solution is to make the assumption that the recognition process will always be unreliable, and compensate by creating the mechanisms for acting robustly at the dialogue management level. In this case, we need to address the limitations described in the previous section. The main ingredients are increasing the ability to detect and diagnose problems, and developing a set of efficient error handling strategies together with the corresponding control mechanisms.

This approach has several advantages. First, automated speech recognition will not become perfect anytime soon. While improvements will most likely happen, the demands will also increase, as the interests will shift towards ever more complex domains and interaction types. This visible trend predicts that recognition performance will not satisfactorily match the tasks at hand in the near future. By default, spoken language interfaces will remain either brittle or significantly constrained. In contrast, systems that can handle the uncertainties at a high level can safely relax the language constraints, and allow more natural interactions over complex domains, even in the presence of imperfect recognition. The approach bears similarities to the ways humans handle uncertainties in communication: after all, human speech recognition capabilities are not perfect either, but we are able to repair conversations and reestablish mutual ground when misunderstandings occur. Finally, although the recognition process is a major source of problems in spoken dialogue systems, it is not the only such source. Errors can be introduced by other components such as language understanding, reference resolution and intention recognition. Increasing the robustness at the dialogue management level will allow spoken dialogue systems to easily accommodate other sources of uncertainties.

The two approaches discussed above – improving recognition accuracy and handling uncertainties at the dialogue management level – do not stand in opposition. Clearly, a combined effort will lead to the best results. The research program proposed in this document is centered on the second approach, and aims to construct the mechanisms for acting robustly at the dialog management level of a spoken dialogue system.

4. Proposed Research Program

4.1. Thesis Statement

The research program proposed in this document is aimed at developing a task-independent, adaptive, and scalable approach for error recovery in spoken dialogue systems. Three properties are pursued in the proposed approach: task-independence, adaptability and scalability.

The approach should be **task-independent**, i.e. the error handling mechanisms should be decoupled from the particularities of the actual dialogue task performed by the system. This separation will allow the reuse of the proposed architecture across different spoken dialogue systems with a minimal amount of authoring effort. Furthermore, it will ensure uniformity and consistency in error handling behaviors within and across tasks.

Secondly, the approach should be **adaptive**. Spoken dialogue systems operate under a large variety of conditions: different performance in the underlying speech recognition and language understanding components, different and changing user populations, different costs for various types of errors, etc. The proposed approach should compensate for these differences by learning from experience how to adapt to the characteristics of particular domains.

Last but not least, the approach should be **scalable**. Previous approaches to learning optimal dialogue policies have shown some success in limited domains, but do not scale to large, practical spoken dialogue systems. The dissertation work proposed in this document aims to overcome these difficulties. The computational costs for learning and acting should grow at most linearly with the size of the dialogue task, making the solution applicable in systems operating with larger, more complex tasks.

4.2. General Approach

To accomplish these goals I propose an approach which decouples the error handling task from the domain-specific control of the dialogue, and regards the former as decision making under uncertainty. The problem is formulated as follows: given relevant information about potential problems in dialogue, make optimal decisions with respect to engaging in one of several conversational error handling strategies. The envisioned solution subsumes three components which, in decision theoretic terms, correspond to constructing the evidence, the actions, and the decision process.

Part I – Indicators (Evidence): The first part of the proposed research program aims at increasing the general awareness of spoken dialogue systems and their capacity to detect and diagnose problems. The goal is to construct a set of *indicators* which can reliably track how well the dialogue is proceeding, and inform the error handling process about potential problems. These indicators will supply the evidence for the error handling decision process.

Part II – Strategies (Actions): The second part of the proposed research program aims at endowing spoken dialogue systems with a richer repertoire of *conversational error handling strategies*. The goal here is to investigate and construct task-independent implementations for an extended set of error handling strategies that can be used to address common problems in human-computer dialogue. These strategies will constitute the set of actions for the error handling decision process.

Part III – Decision Process: Finally, once these two capabilities are in place, the third part of the proposed research program brings them together. The goal is to create a task-independent, adaptable, and scalable *decision process* which can leverage the indicators developed in Part I and the strategies developed in Part II, to make optimal error handling decisions. The desired task-independence and scalability properties will be achieved by decoupling the error handling decision process from the domain-specific control of the dialogue. The use of a data-driven approach for learning the error handling policies will provide the desired adaptability.

4.3. Thesis Plan

Each of the three components of the proposed approach raises many important research questions. Part I has already received a significant amount of attention from the research community. Work in confidence annotation and detecting misunderstandings and corrections has led to the development of several indicators relevant for making error handling decisions. With respect to Part II, a number of conversational error handling strategies, such as explicit and implicit confirmation, disambiguation, etc., have been proposed, investigated and are commonly used. Nevertheless, additional strategies remain to be investigated. Finally, no satisfactory solution exists for the third and central part of the proposed approach: a principled, but at the same time computationally practical framework for making error handling decisions in spoken dialogue systems is still missing. The large majority of systems currently use ad-hoc, heuristic rules for this task. Other approaches based on firmer theoretical grounds have been proposed, but they lack scalability and are not applicable in large, practical spoken dialogue systems. (Section 5 reviews in more detail the relevant work reported to date in the literature.)

The dissertation work proposed in this document will address questions and make contributions in each of these three sub-problems. With respect to Part I, I propose to develop a

novel belief updating mechanism that integrates confidence annotation and correction detection into a unified framework allowing spoken dialogue systems to continuously monitor the reliability of the information they use. With respect to Part II, I propose to construct and investigate task-independent, reusable implementations for an extended set of conversational error handling strategies. Finally, with respect to Part III, I propose to develop a scalable data-driven approach for learning error handling policies.

Concretely, the proposed research program is composed of nine work items, falling under four categories: support work, indicators, strategies and decision process. Three of the nine items represent prior work that already stands completed at the time of this proposal; I will refer to them as *completed work items*. Another four of the nine items represent current and future planned work; I will refer to them as *proposed work items*. Finally, there are two *extension work items*, which I intend to address only if the time constraints will allow it. I now introduce, briefly motivate and describe the goals for each of the nine work items. Table 2 provides a reference summary. Section 6 describes in detail each of these work items.

Support Work

A framework for error handling in spoken dialogue systems cannot be developed and evaluated in the abstract, but rather needs to be an integral part of a larger dialogue management architecture. In this work, the necessary infrastructure will be supplied by RavenClaw [7], a modern dialogue management framework I have developed over the last couple of years. RavenClaw constitutes the first completed work item.

The second completed work item captures the development of several RavenClaw-based spoken dialogue systems to serve as a testbed for evaluation. These systems span several domains and interaction types (i.e. information-access, command-and-control, assistance through procedural tasks). They provide a robust basis for evaluating the proposed architecture, and for validating the task-independence, adaptability and scalability claims.

Part I – Indicators

The first step in the proposed approach is to develop a set of indicators that can reliably track how well the dialogue is proceeding, and inform the error handling process about potential problems.

First, I will develop indicators which measure the reliability of the information that the system handles. The problem amounts to constructing system beliefs over concept values[2]. An accurate confidence annotation mechanism can supply a good starting point for this task. In previous work, captured by completed work item 3, I have addressed this problem, and developed a confidence annotator which leveraged three sources of knowledge in the spoken dialogue system to increase accuracy [12].

Using only the confidence score of the current ASR result to construct beliefs over concept values is however suboptimal, since subsequent events in the dialogue can sometimes carry useful, additional information. In work item 4, I propose to extend my prior work by developing a model which updates the system beliefs in light of the initial confidence scores, as well as follow-up user responses to system actions. This model will integrate confidence annotation and correction detection into a unified framework, allowing spoken dialogue systems to continuously monitor the reliability of the information they use. Further developments addressing the portability of the proposed confidence annotation and belief updating schemes make the subject

[2] The system's belief over a concept will be represented as a probability distribution over the set of candidate values for that particular concept.

of extension work item 5.

SUPPORT WORK Developed a modern dialogue management framework and used it to construct several spoken dialogue systems spanning different domains and interaction-types. The framework and the developed systems will provide the infrastructure for evaluating the proposed error handling approach.	
1. Completed ✓	- RavenClaw Dialogue Management Framework Developed a state-of-the-art dialogue management framework for complex, task-oriented domains.
2. Completed ✓	- RavenClaw-based Spoken Dialogue Systems Developed several RavenClaw-based spoken dialogue systems spanning multiple domains, tasks and interaction types.
PART I – INDICATORS Develop a set of indicators which can reliably track how well the dialogue is proceeding. (These indicators will constitute the evidence for the error handling decision process)	
3. Completed ✓	- Confidence Annotation Created an accurate confidence annotation mechanism by leveraging multiple sources of knowledge in a spoken dialogue system. The confidence scores will provide a starting point for constructing system beliefs over concept values.
4. **PROPOSED**	**- Updating Beliefs over Concept Values** **Develop a model for updating system beliefs over concept values in light of initial confidence scores and subsequent user responses to system actions.**
5. Extension	- Portability of Confidence Annotation and Belief Updating Schemes Study the portability of the confidence annotation and belief updating mechanisms across various tasks, domains and interaction types.
6. **PROPOSED**	**- Non-understanding Indicators and Dialogue-on-Track Indicators** **Construct a set of indicators which carry information about the potential sources of non-understandings (*non-understanding indicators*), and a set of indicators which describe how well the dialogue is proceeding within the scope of a given discourse segment (*dialogue-on-track indicators*).**
PART II – STRATEGIES Develop a large number of conversational error handling strategies. (the strategies will constitute the set of actions for the error handling decision process)	
7. **PROPOSED**	**- Conversational Error Handling Strategies** **Investigate and construct task-independent, reusable implementations for an extended set of conversational error handling strategies.**
PART III – DECISION PROCESS Develop a decision process that can choose among the error handling strategies developed in Part II, based on the indicators developed in Part I	
8. **PROPOSED**	**- Error Handling Decision Process** **Develop a task-independent, adaptive and scalable data driven approach for learning error handling policies in spoken dialogue systems.**
9. Extension	- In-depth Evaluation of Reusability and Adaptability Perform an in-depth evaluation of the reusability of the proposed error handling decision process

Table 2. Summary of work items in the proposed research program

Secondly, in work item 6, I will construct a set of indicators that measure how well the dialogue is proceeding, and carry information about the potential sources of the encountered problems. The assessment and the diagnosis will be performed locally, at the turn level, and globally, at the discourse segment level. For the first case I will construct a set of indicators which characterize the possible sources of turn-level non-understandings (*non-understanding indicators*). For the second case, I will develop indicators which measure how well the dialogue is advancing within the scope of a given discourse segment (*dialogue-on-track indicators*).

Part II – Strategies

The second step in the proposed approach is to construct an extended set of conversational error handling strategies for task oriented spoken dialogue systems. Concretely, in proposed work item 7, I will investigate and develop task-independent, reusable implementations for an extended set of strategies addressing misunderstandings, non-understandings, and other compounded discourse-level problems in human-computer interaction. In the process, I will propose a taxonomy for these strategies, evaluate their general efficiency, and validate the reusability of the constructed implementations.

Part III – Decision Process

The third and last step in the proposed approach is to develop a principled yet computationally practical process for making error handling decisions in spoken dialogue systems. This step is addressed by proposed work item 8. I plan to use a reinforcement learning based approach which can leverage the available indicators and strategies, and learn from experience how to make error handling decisions. Current reinforcement learning based approaches for deriving optimal dialogue policies [52, 54, 57] do not scale to large, practical spoken dialogue systems (see Section 6.4.1). Furthermore, the learned policies are not reusable across domains – a significant amount of human expertise and effort is required for designing models and training them in each new system being developed. In work item 8, I intend to address and overcome these shortcomings by focusing the learning process only on the error handling decisions (rather than on the whole dialogue control problem) and leveraging existing independences between different parts of the dialogue task. Finally, extension work item 9 captures a more in-depth evaluation of the reusability and adaptability of the proposed error handling decision process.

4.4. Thesis Contributions

The goal of the research program proposed in this document is to develop a task-independent, adaptive and scalable framework for error handling in task-oriented spoken dialogue systems.

The main anticipated contributions of this work will be:

- a dialogue management framework for complex, task oriented spoken dialogue systems, which automatically ensures an extended set of basic conversational skills;
- a novel belief updating mechanism which integrates confidence annotation and correction detection into a unified framework, and allows spoken dialogue systems to continuously monitor the reliability of the information they use;
- an investigation of an extended set of conversational error handling strategies for task oriented spoken dialogue systems;
- a scalable data-driven approach for learning error handling policies in spoken dialogue systems operating with large, practical tasks.

5. Literature Review

Some of the problems addressed in this work have already received various amounts of attention from the research community. The following sections review the relevant work reported to date in the literature. These sections are in direct correspondence with the three main parts of the proposed work. Section 5.1 covers research on detecting and diagnosing errors, corresponding to the proposed work on developing relevant indicators for making error handling decisions. Section 5.2 describes work on error handling strategies. Finally, Section 5.3 covers various models proposed to date for making error handling decisions in spoken dialogue systems.

5.1. Detecting and Diagnosing Errors in Spoken Dialogue Systems

Spoken dialogue systems typically rely on confidence information from the speech recognition engine to assess the reliability of their inputs. The confidence annotation problem has traditionally received a large amount of attention. Originally, the efforts were focused on the speech recognition process, and numerous confidence annotation algorithms have been proposed for the frame, phoneme or word level [2, 15, 18]. More recent work has shown that features from other components in the spoken dialogue system can be leveraged to improve accuracy. In [50], San-Segundo et al. used phonetic, language model as well as parsing features with a neural network classifier to successfully detect misrecognized words and out-of-domain utterances. Walker et al [71] constructed a detector for misunderstandings based on decoder, dialogue management and system-specific language understanding features. Van den Bosch et al. [67] focused their attention on a smaller number of features that are available in most spoken dialogue systems and compared two machine learning techniques: RIPPER and memory-based learning. In similar work [12] (also described in Section 6.2.1), we have investigated six different machine learning techniques and leveraged features from the decoder, parser and dialogue manager to construct an accurate confidence annotation mechanism. The general observed result is a consistent 40-60% relative improvement over previous baselines in detecting misunderstandings, confirming that components other than the speech recognizer can provide useful information for the confidence annotation process.

A related problem which has received a fair amount of attention is that of detecting corrections (turns in which the user is trying to correct the system), and aware sites (turns in which the user becomes aware that the system has misunderstood something). In a study based on the TOOT spoken dialogue system [61], Swerts et al find that corrections differ prosodically from non-corrections, and that they are misrecognized by the system more often. In follow-up work, the authors show that both corrections [24] and aware sites [35] can be detected with relatively high accuracy (about 85%) by using features derived from the prosody of the turn, the speech recognition process, and the dialogue history. Similarly, Levow shows that prosodic features can be used to distinguish between initial utterances and later corrections, as well as between corrections of misrecognition errors and corrections of rejection errors [34].

In summary, previous work has shown that misunderstandings and corrections can be detected with significant accuracy by using various machine learning techniques in conjunction with features from different levels in a spoken dialogue system. In the proposed research program (work item 4), I intend to unify these lines of research by constructing a belief updating mechanism which would allow a spoken dialogue system to continuously monitor the reliability of its beliefs, in light of initial confidence scores and subsequent events in the dialogue.

5.2. Conversational Error Handling Strategies

A number of conversational error handling strategies have been previously investigated and are commonly used in various spoken dialogue systems. The two most frequently encountered ones are explicit and implicit confirmation. Both these strategies address potential misunderstandings, and the tradeoffs between them are fairly well understood. Explicit confirmations require an extra user turn, but lead to more predictable user responses, which can be handled more reliably by current spoken dialogue systems. In contrast, implicit confirmations do not slow down the dialogue if they are correct; however, if incorrect, they impose a higher cognitive load on the user, and generally lead to complex responses. Krahmer and Swerts perform an in-depth study of these two strategies [31], identifying a set of the positive and negative cues that people use in response to them, and showing that it is possible to automatically detect corrections from these cues with a fairly high accuracy.

The most frequently encountered strategies for handling non-understandings are repeating the previous system prompt, notifying the user that a non-understanding has occurred, and asking them to repeat or rephrase. Recently, a number of other strategies centered on extracting more information from non-understandings have been proposed. Gorrell et al describe and implement a targeted help strategy based on classifying the non-understood utterance into one of several classes [21]. Raux et al propose a method to automatically generate confirmation prompts that are close to both the non-understood user utterance and covered by the system's language model and grammar [44].

In an effort to identify additional error handling strategies, several researchers have performed wizard-of-oz studies aimed at discovering how humans handle speech recognition errors. In these studies, instead of hearing the actual user utterance, the wizard receives the text of the recognition result, potentially annotated with confidence information. Zollo shows that under these settings wizards tend to provide large amounts of feedback, even when speech recognition works perfectly [75]. Zollo's study also reveals a number of positive and negative feedback strategies: wh-replacement of a missing or erroneous word (e.g. "*how many people are where?*"), attempts to salvage a correctly recognized word (e.g. "*what about the helicopter?*"), explicitly verifying of a turn (e.g. "*you are ready to begin…*"), repeating part or all of the content of the user's utterance, using simple acknowledgements, etc. The majority of the strategies revealed by Zollo's study have no direct equivalent in current spoken dialogue systems. In a similar study in a map domain, Skantze shows that wizards tend to not signal non-understandings, but rather try to advance the dialogue by other means such as asking different task-related questions [58]. It is not yet clear whether the results of Skantze's study apply to other domains as well, or are strongly tied to the particularities of the map-task interaction.

Overall, these studies have confirmed that humans indeed employ a larger repertoire of conversational error handling strategies when faced with uncertainties stemming from poor speech recognition. The second part of the work proposed in this document is aimed closing this gap by investigating and constructing implementations for an extended set of reusable error handling strategies for task-oriented spoken dialogue systems.

5.3. Error Handling Decision Models

In this section I review salient models proposed to date for making error handling decisions in spoken dialogue systems. Subsection 5.3.1 briefly describes several theoretical models of

grounding in human-human communication. In contrast to these theoretical models, most spoken dialogue systems currently rely on heuristic rules to make the error handling decisions; the most frequently encountered heuristic models are discussed in subsection 5.3.2. Finally, the last two subsections review two important recently proposed approaches. Subsection 5.3.3 describes Paek and Horvitz's decision theoretic approach to grounding in human-computer conversation. Last but not least, subsection 5.3.4 describes recent work on using reinforcement learning to derive optimal control policies in spoken dialogue systems.

5.3.1. Theoretical Models of Grounding

In [16] Clark and Schaefer introduced a seminal theoretical model for grounding in human-human communication. The proposed model, dubbed the *Contribution Model*, defines grounding as a collaborative process. Participants in a conversation are contributing to discourse while at the same time trying to reach the following criterion: "the contributor and the partners mutually believe that the partners have understood what the contributor meant to a criterion sufficient for the current purpose" [16]. Each contribution consists of a presentation phase, in which the initiator adds an utterance to the discourse, and an acceptance phase, in which the conversation partner provides appropriate evidence of understanding. The model further describes various classes of evidence of understanding, and four levels at which problems can appear in human-human communication.

In subsequent work, Traum [64, 65] addresses some of the deficiencies of the Contribution Model, and proposes the *Conversational Acts / Grounding Acts* model. Traum's model is an extension of Speech Acts theory which, apart from the core speech acts, includes grounding, turn-taking, and argumentational acts. Seven grounding acts are defined, as well as the protocol capturing their possible sequences. In follow-up work [66], Traum and Dillenbourg propose to use quantitative assessments for the reliability of information and for the costs of actions. To my and the authors' knowledge, no actual implementation of this model has been constructed to date.

Various extensions of the original Contribution Model, as well as other theoretical models of grounding in human-human communication have been proposed [1, 10, 17, 37]. In general, although these theoretical models provide useful insights into how grounding occurs in human-human communication, they are mostly analytical in nature and cannot be directly used to make run-time decisions in spoken dialogue systems. Furthermore, the large majority of these theoretical models[3] lack a quantitative treatment of the reliability of information or of the costs and utilities of various grounding actions.

5.3.2. Heuristic Models for Engaging Error Handling Strategies

In stark contrast to the theoretical models briefly discussed above, most spoken dialogue systems use ad-hoc, heuristic approaches for engaging in error handling strategies.

A certain consensus exists with respect to handling potential misunderstandings. The most often encountered strategies are explicit confirmation, implicit confirmation, rejecting the value for a slot, and accepting the value as grounded. The tradeoffs between these strategies are fairly well understood [31, 40], and the most common models select which strategy to use by comparing the confidence scores against one or two thresholds. Several variations on this theme have been proposed [8, 9, 30, 38, 40, 72]. Some of the existing models use a nominal representation of reliability (i.e. unknown, known, verified, not-verified, etc) [9, 38], others use quantitative assessments based on confidence annotation [30, 40, 72]. The threshold values are

[3] A notable exception is the model proposed by Horvitz and Paek in [43]. The model is discussed in 5.3.3.

set by the system designer, or can be optimized based on assumed [30] or estimated [4] costs for errors.

For non-understandings, the number of possible strategies is larger (see 5.2 and 6.3.1), and the tradeoffs between them are far less understood. The problem is further complicated by the temporal aspect of the decisions which appears when dealing with repeated non-understandings. No generally agreed-upon model exists for addressing this problem. Instead, most systems use handcrafted heuristics, such as asking the user to repeat or rephrase, providing more help, and terminating (or handing-over) the conversation after a number of repeated non-understandings.

5.3.3. Quartet: Conversation as Action under Uncertainty

In [43], Paek and Horvitz introduce a novel decision-oriented, computational formalization for the grounding process in human-machine conversation. The authors regard grounding as decision making under uncertainty and propose a decision theoretic approach which quantifies the beliefs, as well as the costs for taking various grounding actions.

The proposed ideas are materialized in the Quartet architecture [25, 42]. Using Clark's Contribution Model [16] as a basis, and regarding conversation as action under uncertainty, Paek and Horvitz propose an architecture with four interdependent levels for analyzing beliefs and making grounding decisions: Channel, Signal, Intention, and Conversation. The uncertainties within each level are explicitly represented with Bayesian Networks, while the grounding decisions are based on expected utility and value-of-information computations. The Intention level is an exchangeable component which captures the actual task to be performed by the system. The authors illustrate two dialogue systems (Presenter and Bayesian Receptionist) based on the Quartet architecture, using different belief networks in the Intention layer. In DeepListener [26, 27], a spoken command-and-control interface for the LookOut system, Horvitz and Paek use the same decision theoretic approach in a scaled down architecture. The DeepListener system uses dynamic belief networks to accumulate evidence and infer the user's intentions throughout time. The system then selects which clarification strategy or real-world action to engage in by computing the expected utility of each alternative.

In summary, the model proposed by Paek and Horvitz provides a proper computational formalization of the grounding process, and a practical approach for making error handling decisions in spoken dialogue systems, while still retaining a solid theoretical basis. Still, several important shortcomings can be identified: the structure and parameters of the belief networks, as well as the utilities of the various grounding actions are handcrafted. Furthermore, this model has been applied so far only in goal-identification domains (e.g. command-and-control). Because of scalability and task-representation issues, it is not yet clear if the model can be extended to information-access (slot-based) systems, or to the more general family of task-oriented spoken dialogue systems.

5.3.4. Reinforcement Learning for Optimal Dialogue Control Policies

A number of researchers have recently proposed and successfully applied reinforcement learning based approaches to derive optimal dialogue control policies (including the error handling aspects) [33, 51, 54, 57]. In the most common approach, the dialogue management problem (or a subset thereof) is reformulated in terms of a Markov Decision Process (MDP).

Formally, a Markov Decision Process is defined by a finite set of states S, a finite set of actions A, a transition function $T(s,a,s')$ which defines the probability of transitioning to a state s' if the process was in state s and took action a ($Pr(s'|s,a)$), and a reward function $R(s,a)$ which

defines the reward obtained for taking action a in state s [28]. In order to formulate the dialogue control problem as a Markov Decision Process, one must first define the state-space, the set of actions and the reward structure for the model. The state-space is typically constructed by abstracting over the actual dialogue states of the system, in an effort to minimize its size while still keeping it informative. The actions usually correspond directly to system actions, or to a subset thereof. The rewards can be constructed around various dialogue performance metrics, and are typically provided only at the end states of the interactions. Once these three components are defined, learning the optimal dialogue control policy amounts to learning an optimal control policy for the underlying MDP. The latter can be accomplished using standard reinforcement learning algorithms.

The approach has produced successful results in a number of spoken dialogue systems operating with small tasks. Singh et al report on using reinforcement learning to optimize the performance of NJFun, an information access spoken dialogue system [57]. An initial version of the system was used to collect a corpus of exploratory dialogues to serve as training data. The rewards were defined based on a binary measure of task completion, and various choices regarding the type of initiative and confirmation and clarification actions were explored. In the subsequent evaluations of the learned control policy, NJFun showed significant improvements not only in the reward measure for which the optimization was performed, but also on a set of other objective performance metrics. The learned policy outperformed other fixed, handcrafted policies commonly encountered in the literature.

One problematic aspect of this approach is the high cost for collecting training data. An alternative proposed solution is to obtain the data via user simulation. Levin et al [33] derive a simple user model by applying supervised learning on a corpus of dialogues, and then use the model in generative mode to construct the necessary training corpus. Scheffler and Young [52, 53, 54] propose a more sophisticated user model, which explicitly models potential errors as well as constraints to ensure user consistency. The authors compare various state space representations and evaluation functions and successfully apply Q-learning with eligibility traces [52, 53] to optimize dialogue management in a telephone-based movie information system.

The reinforcement learning approach to optimizing dialogue control policies has a number of advantages. First and most importantly, the approach allows spoken dialogue systems to learn optimal behavior from experience and therefore adapt to the characteristics of particular domains and to slow changes in these characteristics. Secondly, the approach can handle delayed feedback, an important feature given the temporal aspects of human-computer interaction. Last but not least, the approach stands on a solid theoretical foundation: a large number of algorithms previously investigated in the reinforcement learning community are available for use.

At the same time, the techniques proposed to date suffer from two major shortcomings which make them impractical in spoken dialogue systems operating with large, complex tasks: (1) the approaches do not scale well with the size of the task to be performed by the system, and (2) once learned, the control policies are not reusable across different tasks and domains. The research program proposed in this document aims to overcome these drawbacks, and construct a scalable, task-independent reinforcement learning approach for the problem of making error handling decisions in spoken dialogue systems. Section 6.4.1 presents an in-depth discussion of these aspects.

6. Proposed Research Program – In Detail

This section describes in detail the proposed dissertation work. The completed, proposed and extension work items are discussed. First, Section 6.1 describes the support work, centered on the RavenClaw dialogue management framework. The following three sections – 6.2, 6.3 and 6.4 – address the three parts of the proposed approach: the indicators, the set of strategies, and the decision process. For each completed work item, the presentation starts with goals and contributions, and continues with a description of the work performed and the results obtained. For each proposed work item, the presentation includes goals and contributions, a description of the proposed approach, the evaluation methodology as well as the anticipated timeline for completing the work.

6.1. Support Work: RavenClaw

▷ 6.1.1. COMPLETED WORK ITEM: RavenClaw Dialogue Management Framework

Goals and Contributions

Developed RavenClaw, a state-of-the-art dialogue management framework for complex task-oriented domains. RavenClaw supplies the necessary infrastructure for the proposed dissertation work. More generally, RavenClaw provides a robust basis for future research on various aspects of dialogue management such as error handling, learning at the task and discourse levels, multi-party conversations, dynamic generation of dialogue plans, etc.

Description

This section gives a brief overview of the RavenClaw dialogue management framework, with a focus on the aspects that are most relevant for the proposed dissertation work. A more thorough description of the framework is available in [7].

The key characteristic of RavenClaw is a separation between the domain-specific and the domain-independent aspects of dialogue control. The domain-specific aspects are encapsulated in a *dialogue task*, essentially a hierarchical-plan for the interaction, constructed by the author of the system. A fixed, domain-independent Dialogue Engine manages the conversation by executing the given dialogue task. In the process, the Dialogue Engine also contributes a basic set of domain-independent conversational strategies implementing error handling, timing and turn-taking behaviors, as well as other universal dialogue mechanisms (help, repeat, cancel, suspend/resume, quit, start-over, etc.) The authoring effort is therefore minimized and focused only on describing the domain-specific dialogue task.

A *dialogue task* is described as a tree of *dialogue agents*, where each dialogue agent handles a subpart of the interaction (e.g. the RoomLine, Login, Welcome, AskRegistered, etc. agents in Figure 1). This hierarchical task representation has several advantages: it scales up gracefully, it is flexible (i.e. it can be dynamically extended at runtime), and it implicitly captures a notion of context in dialogue. The information the system manipulates is encapsulated in *concepts* (slots) [4] stored in the dialogue task tree (e.g. Registered and Name in Figure 1). The

[4] I introduce and use the term *concept* instead of the traditional term *slot*, in order to accentuate the difference between the task-oriented dialogue management framework (such as RavenClaw) and the form-filling dialog management approaches in which slots are typically used. The *concepts* in RavenClaw have a richer representation and set of operators, not commonly encountered in slot-based systems. For more details, see [7]

concepts have a rich representation, which includes: a history of previous values, information about current candidate values and their confidence scores, summary statistics on these confidence scores, information about when the concept was last updated, as well as an extended set of flags describing the concept source, whether or not the concept has been conveyed to the user, whether or not the concept has been grounded, etc.

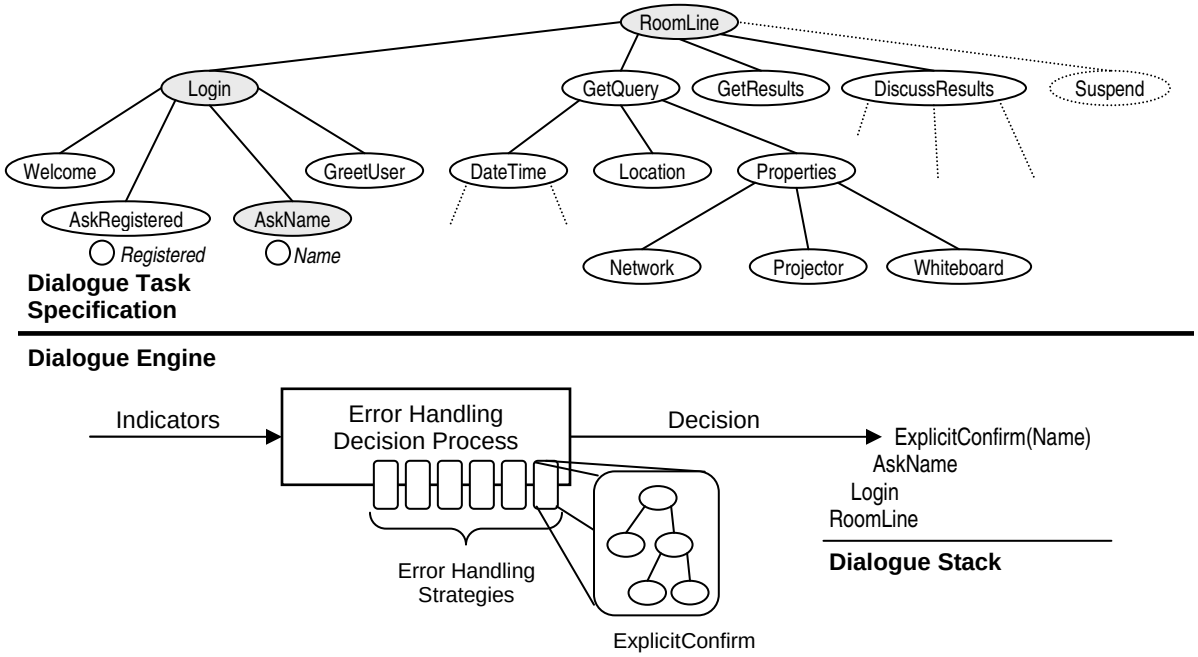


Figure 1. RavenClaw Architecture: the figure illustrates the separation between the dialogue task representation (on top) and the Dialogue Engine (on bottom). The dialogue task is an excerpt from RoomLine, a spoken dialogue system that provides assistance for conference room reservations. The figure also illustrates the operation of the error handling mechanisms in the underlying Dialogue Engine.

The Dialogue Engine algorithms are centered on two data-structures: a ***dialogue stack***, which captures the discourse structure at runtime, and an ***expectation agenda***, which captures what the system expects to hear from the user in a given turn. The dialogue is controlled by interleaving ***execution phases*** with ***input phases***. During the execution phase, dialogue agents from the task tree are placed on and executed from the dialogue stack, creating in the process the system behavior. During the input phase the system uses the expectation agenda to transfer information from the current user input into the concepts defined in the dialogue task tree.

The error handling strategies that will be developed as part of the proposed work (see Section 6.3) are an extension to the set of domain-independent strategies already available in RavenClaw. The decision process which triggers these strategies will be a part of the core Dialogue Engine. This design, in which both the strategies and the decision process are decoupled from the dialogue task as well as from each other, provides several advantages. First, it will ensure that these mechanisms are indeed reusable across different dialogue tasks and domains. Second, it will guarantee uniformity and consistency in error handling behaviors within and across tasks. Third, new error handling strategies will be easily incorporated into any existing

RavenClaw system. Last but not least, the approach will further alleviate the system authoring effort by allowing developers to describe the tasks in a precise, deterministic fashion, without any concerns about the underlying uncertainties in dialogue.

The responsibility for ensuring that the system maintains accurate information and that the dialogue advances normally towards its goals will be delegated to the error handling decision process running in the dialogue engine. The structure and the algorithms governing this process make the subject of Section 6.4. Briefly, at each system turn, this process will assess the available evidence and make a decision with respect to engaging in any of the error handling strategies. The strategies will be implemented as library dialogue agents, using the same hierarchical-plan formalism as the actual dialogue task. When necessary, the error handling decision process will insert an error handling strategy on the discourse stack, dynamically modifying the task which was originally specified by the system's author (see Figure 1). The triggered strategy will become a new part of the system's task, and no other changes are required in the Dialogue Engine to accommodate it. The larger set of domain-independent conversational strategies (e.g. timing and turn-taking behaviors, help, repeat, suspend, etc) will continue to be available throughout the execution of the error handling strategy.

▷ 6.1.2. COMPLETED WORK ITEM: RavenClaw-based Spoken Dialogue Systems

Goals and Contributions

Developed several RavenClaw-based spoken dialogue systems spanning multiple domains, tasks and interaction types. These systems will provide the test bed for evaluating the proposed dissertation work.

Name	Description	Interaction-type	Dimensions
LARRI [6]	Multi-modal spoken dialogue system that provides assistance to F/A-18 aircraft mechanics performing maintenance tasks (developed by D. Bohus, A. Rudnicky, Y. Sun, K. Patel)	Guidance through procedural tasks	# agents: 61+ # concepts: 31+
CMU Let's Go!! Bus Information System [45]	Telephone-based spoken dialogue system that provides an information access interface for Pittsburgh bus routes and schedules (developed by A. Raux, B. Langner)	Information Access	# agents: 44 # concepts: 27
RoomLine	Telephone-based spoken dialogue system that provides assistance for conference room reservations and scheduling in the CMU School of Computer Science (developed by D. Bohus)	Information Access	# agents: 50 # concepts: 20
TeamTalk [62]	A spoken language command-and-control interface for a team of robots; the system is focused on multi-party conversations and asynchronous behaviors in dialogue (developed by T.K. Harris, S. Banerjee, J. Sison, S.P. Kishore, K. Bodine)	Command and Control	# agents: 26 # concepts: 10
Eureka [46]	System that explores the use of dialogue to overcome the difficulties of providing access to a web-search service over the phone (developed by A. Raux)	Information Retrieval	# agents: 47 # concepts: 19

Table 3. RavenClaw-based spoken dialogue systems developed to date

Description

A number of systems have been already constructed using the RavenClaw dialogue management framework. Others are currently under development. Table 3 provides a summary.

The RavenClaw-based systems developed to date span multiple domains and three interaction types: information-access, command-and-control, and assistance and guidance through procedural tasks. These systems provide a solid test bed for evaluating the proposed work and for validating the task-independence and adaptability properties.

Currently, the CMU Let's Go!! Bus Information System is available to the larger public (1-412-268-3526). The call traffic received by the system from outside users as well as developers yields a constant stream of dialogue data. Once deployed, a similar stream of data is expected to be constantly generated by the RoomLine system. These two data streams will form a continuous, background data collection process (BDC). Together with several other focused data collection efforts to be described later in the document, this process will supply the training data for the learning algorithms proposed in this work.

6.2. Indicators

▷ 6.2.1. COMPLETED WORK ITEM: Confidence Annotation

Goals and Contributions

Created an accurate confidence annotation mechanism by leveraging multiple sources of knowledge in a spoken dialogue system. The confidence scores can be used to detect misunderstandings and control the trade-off between different types of errors. They also provide the starting point for constructing system beliefs over concepts values.

Description

In [5, 12], we reported on the development of an utterance-level confidence annotator which leveraged three sources of knowledge in the spoken dialogue system: speech recognition, language understanding, and dialogue management. The problem was cast as a machine learning classification task: given features extracted from these levels, predict whether the input was correctly[5] perceived by the system or not. We trained and compared six different classifiers, in an effort to establish which one was the most effective for the task at hand: AdaBoost, Decision Trees, Bayesian Networks, Support Vector Machines, Neural Networks and Naïve Bayes. With the exception of Naïve Bayes (which performed worse), all the classifiers produced similar improvements. The typical result was 83% accuracy in detecting misunderstandings, amounting to a 50% relative reduction in error rate from the majority baseline, and a 34% relative reduction in error rate from a heuristic rule previously used to detect misunderstandings. This work has also revealed that features from the dialog management and the language understanding components carry valuable information for the confidence annotation process. Later work [5] has shown that a logistic regression model, much simpler than the previous six classifiers, could attain a similar classification performance, and a better soft-error rate.

In follow-up work [4, 5] I proposed a general method for statistically assessing the impact of the various types of errors committed by a confidence annotator on any given dialogue performance metric. The method was applied on data from the CMU Communicator

[5] We regarded the confidence annotator as a detector for misunderstandings; correctness was defined at the semantic level.

spoken dialogue system [49] and the results confirmed the intuitions (e.g. the cost of accepting false information was 1.6 times larger than that of rejecting correct information). The costs derived for the false-positive and false-negative errors allow us to control more accurately the tradeoff between them, so as to maximize dialogue performance in terms of the chosen metric.

In this dissertation work, the confidence annotation mechanism will be used to provide a starting point for constructing system beliefs over concept values.

▶ 6.2.2. PROPOSED WORK ITEM: Updating Beliefs over Concept Values

Goals and Contributions

Develop a model for updating system beliefs over concept values, in light of initial confidence scores and subsequent user responses to system actions. The model will integrate confidence annotation and correction detection into a unified framework allowing spoken dialogue systems to continuously monitor the reliability of the information they use.

Proposed Approach

Initial confidence annotation scores can provide a good starting point for constructing beliefs over concept values. Relying only on confidence annotation scores to accomplish this task is however suboptimal. Subsequent user responses can carry additional information that could be used to adjust confidence in a given concept value. Ideally, spoken dialogue systems should leverage the information available in subsequent events in the dialogue, and continuously update their beliefs throughout time.

The problem is not computationally straightforward. Table 4 gives a couple of examples. Consider the first example. An air travel planning system has already obtained the value "New York" with a confidence of 0.6 for the departure location. Next, the system engages in an explicit confirmation, and the recognition result for the user response contains two hypotheses – "Newark" with confidence 0.6 and "New York" with confidence 0.35. How should the system update its belief following this exchange? The problem can be formulated as follows: given an initial belief (i.e. a probability distribution) over the set of candidate values for a concept, and a set of features characterizing the system's action and the user's response, construct a model that accurately updates the initial belief based on the given features.

Several approaches can be envisioned. The most simplistic one would be to perform a straight-forward probabilistic update by assuming independence between the evidence provided by successive turns. In this case, the computation amounts to multiplying the confidence scores for the different candidate values and renormalizing. This approach has several deficiencies. Consider the second example in Table 4. If the system repeatedly hears a concept value (e.g. "Boston" in the given example), while the dialogue had already moved forward, then it is likely that there was an original misunderstanding that the user is trying to correct, and the recognition process fails repeatedly. Multiplying the confidence scores and renormalizing will boost the confidence in the value heard, while in reality the opposite would be desirable. The approach fails because it makes unrealistic assumptions and it ignores information about the action taken by the system prior to the user's response. Moreover, this approach is not directly applicable when the subsequent recognition result does not contain a value for the concept under consideration (e.g. example 3 in Table 4).

Another possibility is to develop a set of ad-hoc, heuristic rules to govern the update. For instance, if a concept value is explicitly confirmed, we can set its confidence score to 1. If it is disconfirmed, we can set it to 0. A problem with this approach is that user confirmations and

disconfirmations cannot be detected with absolute certainty. Making this type of binary decisions in also inappropriate given the intended use of the constructed beliefs in a probabilistic framework for making error handling decisions.

1.	System belief:	**New York / 0.6**
	System action:	Did you say you wanted to fly out of New York? (**Explicit Confirmation**)
	Decoded result:	**Newark / 0.6 ; New York / 0.35**
	{NewYork/0.6} + Explicit Confirmation + {NewArk/0.6; NewYork/0.35} = ?	
2.	System belief:	**Boston / 0.7**
	System action:	When do you want to leave? (**Next Task Action**)
	Decoded result:	**Boston / 0.75**
	{Boston/0.7} + Next Task Action + {Boston/0.75} = ?	
3.	System belief:	**Austin / 0.7; Aspen/0.2**
	System action:	from Austin… When do you want to leave? (**Implicit Confirmation**)
	Decoded result:	**[_non_understanding_] + BargeIn**
	{Austin/0.7; Aspen/0.2} + Implicit Confirmation + {}/BargeIn = ?	

Table 4. Three examples of the belief updating problem: given an initial system belief, a system action, and the recognition results for the user's response, how should the system update its belief about the departure location?

Instead, I propose a data-driven approach. Two potential models are described below.

Model 1: **Updating Beliefs over Top-N Candidate Values in a Dynamic Belief Network**

This first proposed model updates the beliefs over the top-N candidate values for a concept in a dynamic belief network.

The model structure will be fixed a priori (see Figure 2). The variable of interest, *C*, models the belief over the concept values at times *t* and *t+1*. The node *C* at time *t* models the initial system belief. The node *C* at time *t+1* models the true state of the concept, which is the same as the desired updated system belief. The evidence nodes correspond to features describing the system's last action, and the user's response. The user response features capture both confidence annotation information and aspects related to correction detection: the presence of positive or negative lexical markers, the parse quality and utterance length, the presence of repetitions or of new information, etc. The evidence corresponding to the user response is conditioned on both the system action and on the true state of the concept *C*, captured by the target belief at time *t+1*.

The parameters of the model will be learned from data. The approach requires a labeled corpus, with training instances containing the current belief, the system action, the user's response, as well as the actual concept value (used to construct the target belief for concept *C* at time *t+1*).

A model which tracks the belief over all the possible values for a concept *C* would be most desirable. This approach is however intractable for concepts with large numbers of possible values such as city names, date and time concepts, numbers, etc. The massive number of parameters in such a model would render any learning and generalization impossible. Instead, I propose an alternative in which we abstract over the actual set of possible values for a concept *C*, and estimate a belief for the current top-*N* candidate values for that concept, without regard to what those top-N values exactly are. *N* is a small fixed number. In practice, it is arguably

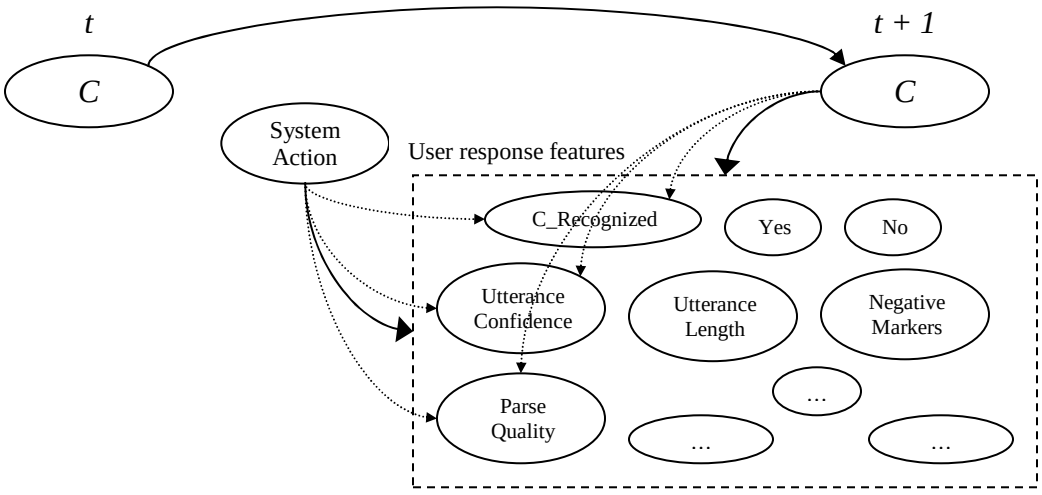


Figure 2. Potential structure for a dynamic belief network which models belief updates over the top-N candidate values for a concept (only a subset of the user response nodes are illustrated)

sufficient to track beliefs over 3 to 5 candidate values. The technical aspects of the anticipated abstraction mechanism are discussed in more detail in Appendix B.

Model 2: Updating Individual Confidence Scores Using Logistic Regression

The second proposed model does not update beliefs (i.e. probability distributions) over candidate values, but rather the individual confidence scores of each candidate value. In this approach, the update problem can be formulated as a logistic regression task: given an initial confidence score c_v^t for a certain candidate value v at time t, and a feature vector $\overline{f}$ characterizing the system action and following user response, construct an updated confidence score c_v^{t+1}.

This model updates the confidence scores for each candidate value independently, and as a result the scores will no longer form a belief (i.e. a normalized distribution over possible values). For instance, in the third example from Table 4, the logistic regression model will be applied independently to the "Austin" and "Aspen" candidate values to determine their updated confidence scores. This approach is no longer limited to tracking only the top-N values, since the logistic regression model can be instantiated repeatedly for each candidate value. The advantage comes at the price of making an unrealistic assumption of independence between the candidate values.

The negative effects of this assumption can be potentially counteracted by adding features which summarize the set of other candidate values for that concept, as well as the relationship between the value addressed by the model and the values in that set. For instance, in updating the confidence for a certain candidate value v, we can consider: the number of other candidate values for the same concept at that point in time, whether or not v has the top confidence score, whether the last system action explicitly mentioned v or one of the other values, etc. A careful design of the feature set is required, but the potential benefit is a simpler model with no constraints on the number of candidate values which can be handled at any given time.

Evaluation

The performance of the proposed belief updating model will be evaluated in terms of hard-error rate (accuracy) and soft-error rate (average log-posterior of the correct value). The developed model will be compared against the baseline performance of simple heuristic update rules, such as the ones commonly encountered in today's spoken dialogue systems.

Timeline (see Appendix A)

The timeline for completing this proposed work item includes:

- Perform a data collection experiment (DC-1) aimed at constructing the corpus necessary for training and evaluating the model. The anticipated data collection effort will start once the concept-scoped strategies (see Section 6.3) are operational (4 months, timeline item 5);
- Develop, evaluate and fine-tune the proposed models; integrate them in the RavenClaw dialogue management framework (5 months, timeline item 6);
- Write a paper describing the developed belief updating mechanism (3 months, timeline item 8).

▷ 6.2.3. EXTENSION WORK ITEM: Portability of Confidence Annotation and Belief Updating Schemes

An important issue that remains to be addressed in the context of the confidence annotation and belief updating mechanisms is how well they transfer across different spoken dialogue systems.

An assessment of the portability of the developed mechanisms can be easily performed by training them with data from one domain and testing their performance on data from another domain. Conditioned on the nature of the observed results, follow-up work will be aimed at improving performance in this respect. The goal is to transfer the constructed confidence annotation and belief updating mechanisms from one domain to another, and optimize their performance in the new domain with a minimal amount of effort. To this end, I intend to explore techniques such as co-training and self-training [29, 39], which can successfully leverage small amounts of labeled data and larger amounts of unlabeled data from a new domain to increase performance.

▶ 6.2.4. PROPOSED WORK ITEM: Non-Understanding and Dialogue-on-Track Indicators

Goals and Contributions

Construct a set of indicators which carry information about the potential sources of non-understandings (*non-understanding indicators*), and a set of indicators which measure how well

the dialogue is proceeding within the scope of a given discourse segment (*dialogue-on-track indicators*).

Proposed Approach

Non-Understanding Indicators

A non-understanding is said to occur when the spoken dialogue system is not able to extract any meaningful information from the semantic representation of the user's utterance (see turns 6, 14, 16 in the sample conversation from Section 1). This mode of failure can be triggered by a large variety of problems at different levels in the system. Speech recognition problems, such as background noises, speaker variability, out-of-vocabulary words, can lead to errors which can ultimately result in non-understandings. In some cases, even if the recognition is successful, out-of-domain utterances and insufficient grammar coverage can generate similar problems at the language understanding level. Finally, even when both the recognition and parsing processes function correctly, the dialogue manager may fail to integrate the information in the user's turn if that information falls outside the scope of the conversation as defined by the system at a given point. In this last case, the problem appears at a higher, pragmatic interpretation level, but it still manifests itself as a non-understanding.

In general, spoken dialogue systems know with certainty when non-understandings occur in dialogue – there was a user turn, but no meaningful information could be extracted from it. A binary non-understanding indicator is therefore directly available in most systems. In this work I propose to construct a set of additional indicators which carry information about the potential sources of the non-understandings. These indicators will provide the evidence for engaging in various strategies for handling non-understandings, such as asking the user to repeat or rephrase, lexically entraining the user, switching the input modality, etc. (see Section 6.3, and Table 6).

Name	Level	Description
RecoConf	Recognition	The confidence score of the recognition
NoWords	Recognition	Indicates whether the recognition result contains any words (or just acoustic non-lexical events)
SNR	Recognition	Signal-to-noise ratio for the channel
NoParse	Parsing	Indicates whether or not the language understanding layer formed a semantic representation of the input
ParseFragmentation	Parsing	Measures the parse quality in terms of fragmentation
ParseCoverage	Parsing	Measures the parse quality in terms of how much coverage it provides for the user's utterance
BlockedMatch	Interpretation	In case there is a parse, indicates that the non-understanding occurred because the semantic input matched a system expectation that was currently blocked (i.e. the system was not open to talk about that subject at that point in time)
NoMatch	Interpretation	In case there is a parse, indicates that the non-understanding occurred because the semantic input did not match any system expectation

Table 5. Some non-understanding indicators.

Table 5 illustrates some of the indicators currently under consideration. With the exception of the signal-to-noise ratio (SNR), these indicators are already available in the RavenClaw dialogue management framework. In the future, I will concentrate on identifying and implementing other informative indicators, at each of these three levels: recognition, language understanding (parsing) and dialogue management (interpretation).

Dialogue-on-Track Indicators

The dialogue-on-track indicators will measure how well the dialogue is advancing towards its goals within each discourse segment or topic. These indicators will provide the evidence for engaging in topic-level error handling strategies, such as restarting a topic, switching to an alternative dialogue plan, etc. (see Section 6.3, Table 6)

A first group of proposed dialogue-on-track indicators consists of topic-level summary statistics for each of the non-understanding indicators discussed above (e.g. counts, averages, etc.) A second group is represented by indicators which characterize the rate at which the dialogue advances towards its goals within a given discourse segment. Examples of potential indicators in this second group include: the ratio between task-level dialogue actions and error handling dialogue actions, the number of current consecutive error handling actions (error spiral depth), etc.

The anticipated work will focus on defining these indicators and implementing the necessary support for computing them at runtime in the RavenClaw dialogue management framework.

Timeline (see Appendix A)

The timeline for completing this proposed work item includes:

- Implementation of the proposed indicators (3 months, timeline item 7).

6.3. Strategies

▶ 6.3.1. PROPOSED WORK ITEM: Conversational Error Handling Strategies

Goals and Contributions

Investigate and construct task-independent, reusable implementations for an extended set of conversational error handling strategies.

Proposed Approach

The proposed plan for this work item consists of three steps: define the strategies, implement them, and evaluate their performance.

Step 1: Define strategies (partially completed)

The first step is to identify a large set of conversational error handling strategies which can be used to address frequent problems in human-computer dialogue. The existing literature provides a good starting point for this task (see Section 5.2). Furthermore, an analysis of errors in human-computer dialogue corpora may lead to additional alternatives.

Tables 6 and 7 present the list of strategies that are currently under consideration. The strategies are grouped in several classes. The level of abstraction introduced by the proposed classification supports various architectural design choices in the error handling decision process (see Section 6.4), and provides a framework for identifying and accommodating new strategies to

be proposed in the future.

A first characteristic separating the strategies into two major classes is whether they are initiated by the system or by the user. A number of system-initiated strategies are presented in Table 6; they will constitute the set of actions for the error handling decision process. The user-initiated strategies, illustrated in Table 7, will be available as error handling conversational skills that the system can engage in at the user's request.

The system-initiated error handling strategies have two main responsibilities: (1) to ensure that the system operates with correct information and (2) to ensure that the dialogue is on track and is advancing normally towards its goals. Depending on which one of these two goals they address, the system-initiated strategies can be further clustered into two classes (see 1st column in Table 6).

Strategies in the first class aim to ensure that the system operates with correct information. They address potential misunderstandings, i.e. situations when the system has already acquired some information, but is uncertain about its reliability. These strategies operate on individual concepts, and include explicit and implicit confirmation, disambiguation, asking the user to repeat the value of a concept, and rejecting the value of a concept.

Strategies in the second class aim to ensure that the dialogue is on track and is advancing normally towards its goals. A first subset of these strategies addresses local, turn-level non-understandings, i.e. situations in which the system fails to extract any meaningful information from the user's turn. This subset includes strategies like switching the input modality, asking the user to repeat or rephrase, notifying the user that a non-understanding has occurred, providing more help, etc. A second subset of strategies in this class is geared towards global, discourse-level compounded problems, and includes strategies such as restarting a topic, switching to an alternative dialogue plan, handing-over the interaction to an operator, etc.

Finally, the dialogue entity addressed by each strategy – concept, turn, or topic (see 2nd column in Table 6) – provides a third dimension for classifying the strategies, and informs various design choices in the error handling decision process (see Section 6.4).

The functionality of each strategy will be defined in terms of the system behavior as well as the expected user behaviors during the execution of the strategy. The second part is very important. Spoken dialogue systems must know how to engage in error handling strategies, but also how to handle the follow-up user responses. Certain strategies, such as implicit confirmation, disambiguation, etc. can trigger complex user reactions [31], and have the potential to lead the dialogue into a spiral of errors. The system must be able to constrain or correctly anticipate the common user responses within each of the proposed error handling strategies.

Step 2: Implement (partially completed)

The second step is to implement the selected strategies in the RavenClaw dialogue management framework. The anticipated implementation effort will be dominated by two aspects: the first one is implementing the desired functionality in a task-independent fashion; the second one is constructing the necessary support in the language understanding and language generation components of the system.

10 of the 30 strategies currently under consideration have been already implemented in the RavenClaw dialogue management framework (see Tables 6 and 7). Work is in progress on 4 others. For the rest of the strategies, the last column in Tables 6 and 7 indicates the anticipated difficulty of the implementation effort.

Goal Type of problem addressed	**Dialogue Entity**	**Strategy Name** ***Example***	**Implementation**	
			Status	**Expected difficulty**
Ensure that the system operates with reliable information misunderstandings	Concept	**Explicit Confirmation** *Did you say you wanted to fly from Pittsburgh?*	Finished	
		Implicit Confirmation *from Pittsburgh … When do you want to fly?*	In progress	Difficult
		Disambiguation (*) *Did you want to leave from Boston or Austin?*		Difficult
		Ask Repeat Concept (*) *Can you tell me again where were you flying from?*		Easy
		Reject Concept *I'm sorry I don't think I understood the city name*		Easy
Ensure that the dialogue is on track, and advancing normally towards its goals non-understandings and problems at the discourse level	Turn	**Switch Input Modality** *Can you please dial that number on the phone pad?*		Easy
		SNR Repair *I think there is a lot of noise on the line. Can you please move to a quieter environment?*		Moderate
		Ask Repeat Turn *Can you repeat that please?*	In progress	Easy
		Ask Rephrase Turn *Could you please rephrase that?*	In progress	Easy
		Notify Non-understanding *I'm sorry, I'm not sure I understood you correctly*	Finished	
		Explicit Confirm Turn *Did you mean you wanted to fly tomorrow?*		Difficult
		Targeted Help (*) *To make a hotel reservation, you could try something like "I'd like a hotel in San Francisco"*		Difficult
		WH-Reformulation (*) *You said you wanted a hotel where?*		Difficult
		Keep-a-word Reformulation (*) *What about Pittsburgh?*		Difficult
		Generic Help *Here are some tips for a smooth interaction… Please speak clearly and naturally. If you wish to make a correction just restate the new information …*		Easy
		You Can Say … *You can say a city name, followed by a state name, like 'Austin, Texas'. To hear other things you can say at this point, say 'More'.*	In progress	Easy
	Topic	**Restart (Sub)Task Plan** *Okay, I'm sorry I'm a bit confused. Let's try this part again….*		Difficult
		Select Alternative (Sub)Task Plan *Okay, let's try this in a different way. Can you first tell me the state?*		Moderate
		Start Over *I'm sorry I am a bit confused. I'm afraid we will have to start over.*	Finished	

		Terminate Session / Hand Over *I'm sorry I got a bit confused. I will transfer you to an operator now.*		Easy

Table 6. System-initiated conversational error handling strategies; the strategies marked with a star (*)will be implemented only if the time constraints will allow it.

Strategy Name ***Example***	**Implementation**	
	Status	**Expected difficulty**
Help *Help!*	Finished	
Where Are We *Where are we?*	Finished	
Start Over *Can we start from the beginning?*	Finished	
Scratch Concept Value *You've got the wrong city name!*		Difficult
Go Back *No, I didn't mean that… Go back!*		Difficult
Channel Establishment *Hello! Hello! Are you still there?*	Finished	
Suspend / Resume *Suspend!*	Finished	
Repeat *Can you say that again?*	Finished	
Summarize *Summarize!*		Moderate
Quit *Bye bye!*	Finished	

Table 7. User-initiated conversational error handling strategies

Step 3: Analyze and Refine

The third and last step in this work item is to analyze the performance of the implemented strategies, and bring further refinements in light of the observed results.

First, I will validate the task-independence and reusability claims for the constructed error handling strategies. This goal will be achieved by deploying the strategies in different RavenClaw-based spoken dialogue systems. Potential problems and limitations will be identified by analyzing the dialogue corpus collected throughout the background data collection process (BDC) and the first set of data collection experiments (DC-1). In light of the observed results, I will make the necessary adjustments and refinements in the definitions and implementations of the proposed strategies.

Secondly, I plan to analyze the efficiency of the turn-level non-understanding strategies[6]. It is important to note that the efficiency of a strategy cannot be defined in general, but rather depends on the decision process that is used to engage it – a strategy will clearly be more efficient if engaged exactly when needed (i.e. when the problem that the strategy addresses is the one currently affecting the interaction) [7]. Nevertheless, we can establish lower and upper bounds on the efficiency of these strategies.

A lower bound for the efficiency of each non-understanding strategy will be constructed by analyzing these strategies in conjunction with a decision process that engages them uniformly. This first evaluation will be based on data gathered in the DC-1 experiments (see Appendix A), and is expected to reveal rough, "on average" differences between the various non-understanding strategies. For instance, a strategy which lexically entrains the user is expected to be more efficient on average than one which merely repeats the previous system prompt. In addition, I plan to construct a "gold standard" [41] for the efficiency of these strategies by performing a wizard-of-oz experiment in which the decision process will be performed by a human operator. The wizard will have full knowledge of the user's utterances, and of the system's perception of the user utterances (e.g. recognition results, various indicators, etc). This second evaluation will provide an upper bound for the performance of each non-understanding strategy. At the same time, it will serve as a checkpoint in validating the hypothesis that the proposed non-understanding strategies can indeed increase the robustness of spoken dialogue systems.

Evaluation

See Step 3 of the proposed plan for this work item.

Timeline (see Appendix A)

The timeline for completing this proposed work item includes:

- Complete the implementation of the misunderstanding and non-understanding error handling strategies in the RavenClaw framework (4 months, timeline item 1);
- Evaluate non-understanding strategies in conjunction with a decision process which engages them uniformly, and in the described WOZ setting. Implement the reminder of the strategies (6 months, timeline item 4)
- Write a paper describing the task-decoupled implementation of conversational error handling strategies in the RavenClaw dialogue management framework (3 months, timeline item 3).

6.4. Error Handling Decision Process

▶ 6.4.1. PROPOSED WORK ITEM: Error Handling Decision Process

Goals / Contributions

Develop a dialogue-task-independent, adaptive, and scalable data driven approach for learning error handling policies in spoken dialogue systems.

[6] A simple metric which can be used to measure efficiency is the percentage of times that the next turn contains another non-understanding. Other metrics, involving the lengths of the error segments can also be envisioned.
[7] Other factors, such as interaction type, user population, etc. might also affect the efficiency of each strategy

Proposed Approach

To accomplish these goals, I propose to use a reinforcement learning based approach. Reinforcement learning has already been used successfully to derive optimal dialogue control policies in several spoken dialogue systems operating with small tasks [33, 52, 54, 57] (see Section 5.3.4 for a brief review). One of the main advantages of this approach is that it allows a system to learn from experience and therefore adapt to the characteristics of particular domains, and to slow changes in these characteristics. Furthermore, the approach allows learning from delayed feedback, an important feature given the temporal aspects of human-computer interaction.

The reinforcement learning approaches proposed to date suffer however from one important shortcoming, which so far has prevented their use in large, practical spoken dialogue systems. The problem is lack of scalability. The main difficulty arises from the design of the state-space of the underlying Markov Decision Process. The number of parameters to be estimated in the model (the transition probability matrix) grows quadratically with the number of states. The number of states grows exponentially with the number of state variables, which in turn grows linearly with the size of the task to be performed. In spoken dialogue systems operating with larger tasks, this leads to an explosion of the state-space which makes the learning problem intractable. To alleviate this problem, an abstraction of the dialogue state is often used to design the underlying MDP, at the cost of potentially violating the Markovian assumption. This type of abstraction can yield a linear growth of the state-space with the size of the task, but we are still faced with a quadratic growth of the number of parameters and ultimately training data requirements.

Consider as an example the NJFun spoken dialogue system [57]. The system operated with 3 concepts (attributes). The state space was defined by 7 state variables, with one variable capturing the attribute the system was currently working on. This resulted in state-space size of 62. A dialogue control policy was learned from 311 dialogue sessions. Consider now a spoken dialogue system which operates with 12 instead of 3 concepts[8]. Using the same dialogue-state abstraction, the resulting MDP would have 242 states in this case. Although the size of the state-space increased only 4 times, the number of parameters to be estimated in the model, and hence the amount of data needed to converge on a policy grows 16 times. Given the difficulties and high costs associated with obtaining training data for spoken dialogue systems, this approach becomes quickly impractical when applied to systems operating with larger, practical tasks.

A second shortcoming of the reinforcement learning techniques proposed so far is that the learned dialogue policies cannot be reused across tasks. This problem also stems from the design of the underlying MDP. The state and action spaces are constructed starting from the actual dialogue states and actions, and retain task-specific aspects. For each new system being built, a new MDP has to be constructed, new data has to be collected, and a new training phase is necessary. The learning approach is aimed at automatically deriving an optimal dialogue policy while minimizing the development effort, but in reality quite a large amount of expertise and effort are still required at development time.

The dissertation work proposed in this document aims to overcome these drawbacks. The goal is to develop a reinforcement learning based solution to the problem of making error handling decisions. The solution should be task-independent, scalable and should favor reusability of the learned policies across different domains.

[8] Some of the RavenClaw-based spoken dialogue systems have a much larger number of concepts: RoomLine uses 20, CMU Let's Go!! Bus Information System uses 27, LARRI uses over 31 (the task grows dynamically)

The anticipated solution is based on two ideas.

1. Decouple the error handling decisions from the task control decisions and focus the reinforcement learning process only on the error handling policies;
2. Leverage the independences that exist between different subparts of the dialogue task.

The first idea is to concentrate the reinforcement learning process only on learning error handling policies, as opposed to all the dialogue control decisions. Traditionally, reinforcement learning approaches have been aimed at deriving the whole dialogue policy, which includes both error handling and task-specific control aspects. For instance, the action set in NJFun [57] captured both error handling actions, such as explicitly confirming an item, and task-specific actions, such as greeting the user and providing the results; similarly, the state-space included a variable indicating whether the user had been greeted or not. In this work, I propose to decouple the error handling and task-specific control aspects of the dialogue, and use reinforcement learning to derive only the error handling policy. The task-specific control decisions will be handled separately by a purely deterministic dialogue management framework, such as the hierarchical-plan based approach in RavenClaw[9].

This approach has several advantages. First, it simplifies the reinforcement learning problem: the state and action spaces become smaller since we are no longer concerned with task-specific control decisions. Second, the learned policies are more likely to transfer to new tasks and domains, since they are decoupled from the actual dialogue task to be performed by the system. Last but not least, the decoupling will further lessen the development effort by allowing system authors to focus their attention only on the task-specific aspects of the dialogue, while not concerning themselves with the underlying uncertainties and error handling aspects.

The second central idea is to make use of the independence relationships that exist between different parts of a dialogue task. For instance, in a flight reservation system, the mechanisms by which the system ensures the correctness of the departure city can be safely assumed to operate independently of the status of the other slots (e.g. departure time, arrival city, etc). Most often, the error handling mechanisms can operate on a local basis, independent of what happens in other subparts of the dialogue. I intend to leverage this type of independence assumptions, and use a "divide and conquer" approach to make the reinforcement learning techniques tractable in practical, complex spoken dialogue systems.

I will now describe in detail the proposed decision process, based on the ideas outlined above.

Gated Markov Decision Processes

Figure 3 illustrates the overall structure of the proposed error handling decision process.

A separate Markov Decision Process (MDP) is associated with each concept and each topic (i.e. dialogue agent, i.e. node) in the dialogue task tree[10]. The role of *concept-MDPs* is to ensure that their corresponding concept holds correct information. The action-space for these models spans the concept-level strategies (see Table 6), such as explicit confirmation, implicit

[9] The idea of decoupling task specific control decisions from error handling decisions and focusing the reinforcement learning on the latter also appears in Konrad Scheffler's Ph.D. thesis on automatic learning of dialogue strategies [52] as a subject for future work.

[10] Note that both the figure and the following discussion make reference to the RavenClaw dialogue management framework. The proposed approach can nevertheless be implemented in any dialogue management framework that exposes a notion of concept (i.e. slot, attribute), and a notion of topic (i.e. discourse segment).

confirmation, disambiguation, etc. The role of the *topic-MDPs* is to ensure that the dialogue advances normally towards its goals within each topic (discourse segment). The topic-MDPs that correspond to request dialogue agents (terminal nodes in the dialogue task tree[11]) handle the turn-level non-understanding strategies, while the topic-MDPs running on non-terminal dialogue agents handle the topic-level strategies.

At each turn in the dialogue, all the MDPs are polled for their decisions. A gating mechanism [36] is then used to arbitrate between the actions suggested by the individual MDPs, and decide which action will be ultimately performed by the system.

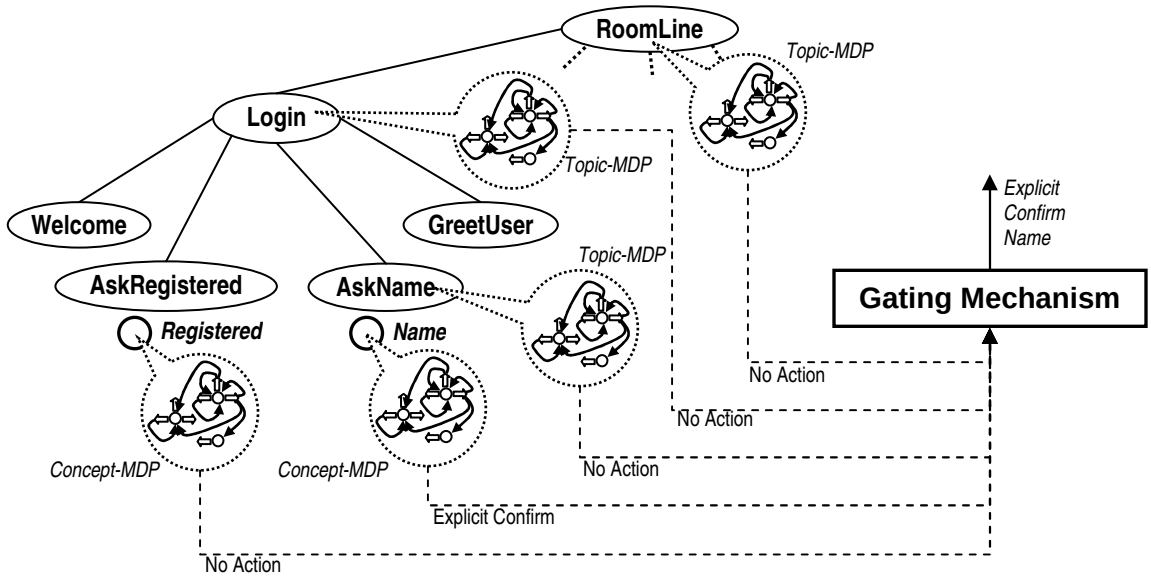


Figure 3. Structure of the proposed Gated-MDP model. At each time step, each of the MDPs running on the concepts and topics in the dialogue task tree forwards at their suggested action to a gating mechanism, which ultimately decides which action will be performed by the system.

The proposed approach has several advantages. First, the size of the concept-MDPs and topic-MDPs can be maintained relatively small. The structure of the models (in terms of state and action spaces) is the same across all concept-MDPs and across all topic-MDPs. As the complexity of the task grows, the number of MDPs grows, but the dimensions and structure of the individual models remain fixed. A more complex task means a larger number of models (or model instantiations), rather than a more complex model structure.

Second, not only the structure, but also the parameters (transition probabilities) of the MDPs can be tied across models. For instance, in a flight reservation system, the model responsible for grounding the departure city concept should probably behave in the same manner as the model responsible for grounding the arrival city. The decision to tie models is best informed by knowledge and constraints from the domain, and is left to the system author. Tying the models has the potential to greatly improve scalability, since the actual number of parameters that need to be learned will grow sub-linearly with the size of the task. Model tying

[11] In RavenClaw, the request dialogue agents occupy terminal (leaf) positions in the dialogue task tree. Their role is to ask the user a question (or present some information) and then give the turn to the user.

also permits a certain degree of control over the complexity of the overall model, which can be adjusted to match the available training data.

Third, expert knowledge can be used to provide a reasonable starting point for the error handling policies. The initial policies for the underlying MDPs can be easily handcrafted such as to produce sensible behaviors throughout the exploration phase of the learning.

Fourth, the proposed approach favors reusability. Since the individual MDPs are decoupled from the actual dialogue task performed by the system, the policies learned within a certain dialogue task can potentially be reused in other tasks. In the least, the policies derived in one task can serve as good starting points for exploration and learning in a new task.

Last but not least, the approach can easily accommodate dynamic dialogue task generation. Traditional RL-approaches cannot be applied when the dialogue task is not precisely known in advance, since the underlying MDP has to be designed based on the dialogue task. The proposed approach circumvents this problem by decoupling the structure of the concept- and topic-MDPs from the actual dialogue task. The dialogue task tree can be extended dynamically at runtime, and new instances of concept- and topic-MDPs can be added on the fly.

A potential problem of the proposed approach is that by running separate decision processes for each concept and topic in the dialogue, we are assuming independence between the error handling decisions across these entities. While this assumption is reasonable in a large number of cases, it is likely that sometimes better decisions could be made by taking into account a larger context. This problem can potentially be alleviated by including other global, but still task-independent variables in the state-space design of the underlying MDPs.

Next, I discuss in more detail each of the major components of this decision process: the gating mechanism, the design of the concept- and topic-MDPs, the reward structure, and choice of learning algorithms.

Gating Mechanism

The gating mechanism arbitrates between the set of actions suggested by the concept- and topic-MDPs at each time step. This arbitration process is necessary since in general the system cannot engage in more than one error handling strategy simultaneously.

I anticipate that a simple heuristic gating mechanism derived from domain-independent dialogue principles will suffice. For instance, the gating mechanism could prioritize the concept- and topic-MDPs by giving preference to topics over concepts, and generally to topics and concepts that are closer to the current focus of the conversation. More sophisticated mechanisms might include more information from the run-time discourse structure, as well as information about the identity of the actions, or their utilities in the underlying MDPs.

An alternative option would be to attempt to learn an optimal gating function from data. The problem amounts to optimally coordinating a set of MDPs, and has recently received some attention in the reinforcement learning community [3]. Given however the large amounts of knowledge about the gating behavior that can be inferred from first dialogue principles, I consider the heuristic approach more practical and justified in this case.

Structure of concept-MDPs and topic-MDPs

The structure of the underlying concept- and topic-MDPs will be dictated by the indicators and strategies constructed in Parts I and II of the proposed dissertation work (see Sections 6.2 and 6.3). The state spaces will be defined by an informative subset of the cartesian product of the selected indicators. The actions will correspond directly to the available system-initiated conversational error handling strategies which operate on concepts, turns and topics

(see Table 6). I briefly discuss a couple of anticipated designs below (they are subject to change upon further investigation).

State variables
C (confidence level):
VL (very low, c < 0.2)
L (low, 0.2 < c < 0.4)
M (medium, 0.4 < c < 0.6)
H (high, 0.6 < c < 0.8)
VH (very high, 0.8 < c)
V (value)
1 – at least a candidate value exists
0 – no candidate value exists

Actions
EC – Explicit Confirmation
IC – Implicit Confirmation
R – Reject Concept
NoA – No Action

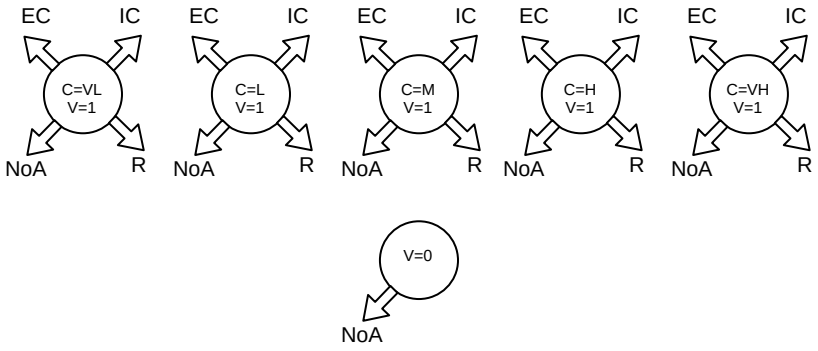


Figure 4. Sample structure for a concept-MDP: the model has 4 actions and 6 states corresponding to five levels of confidence on the top candidate value and whether or not there is a candidate value (for clarity, the transitions are not illustrated; they go from every action to every state).

The concept-MDPs aim to ensure that the system has the correct value for each concept. Their actions are the concept-level strategies: explicit confirmation, implicit confirmation, disambiguation, rejection. To these, we add "No Action" to model the case when the model does not engage in any strategy. A set of relevant indicators includes: the confidence of the top candidate value, the number of candidate values, the difference in confidence between the top and second candidate values, how long ago the concept was acquired, etc. Figure 4 illustrates a very simple design for the concept-level MDP, which takes into consideration only the confidence of the top candidate value (binned into 5 categories), and excludes the disambiguation strategy. More complex designs which also take into account more state variables along the lines described above and include all the proposed concept-level error handling strategies can be easily envisioned.

For the topic-level MDPs, I anticipate using two types of models.

A first type of model will be associated with each of the request agents in the dialogue task tree. These models will address turn-level non-understandings, and their action-space subsumes the turn-level non-understanding strategies (see Section 6.3.1): asking the user to repeat or rephrase, lexically entraining the user, providing more help, etc. The design of the state-space for these models will use the various non-understandings indicators developed in Part II (see Section 6.2.3).

The second type of model is associated with each of the non-terminal nodes in the dialogue task tree. These models monitor the dialog progress at the topic level, and engage in the

topic-level error handling strategies (see Section 6.3.1.): restarting a topic, switching to an alternative dialogue plan, etc. The design of the state space for these models will use the various dialogue-on-track indicators developed in Part II (see Section 6.2.4).

The final shape of the concept and topic-level MDPs is still under investigation. I anticipate exploring different designs, based on the indicators and strategies that will be available. Other considerations include adding non-local indicators to alleviate the potential negative effect of the independence assumptions made.

Reward structure

Two options are available for assigning rewards in the gated MDP model (see Figure 5). The first option is to provide the rewards in a global fashion, over the concerted action of the whole model, as illustrated in 5.a (post-gating reward). The second option is to define the rewards locally, for each individual MDP, as illustrated in 5.b. Both schemes are currently under investigation. I now briefly discuss some of the advantages and disadvantages of each of the two schemes.

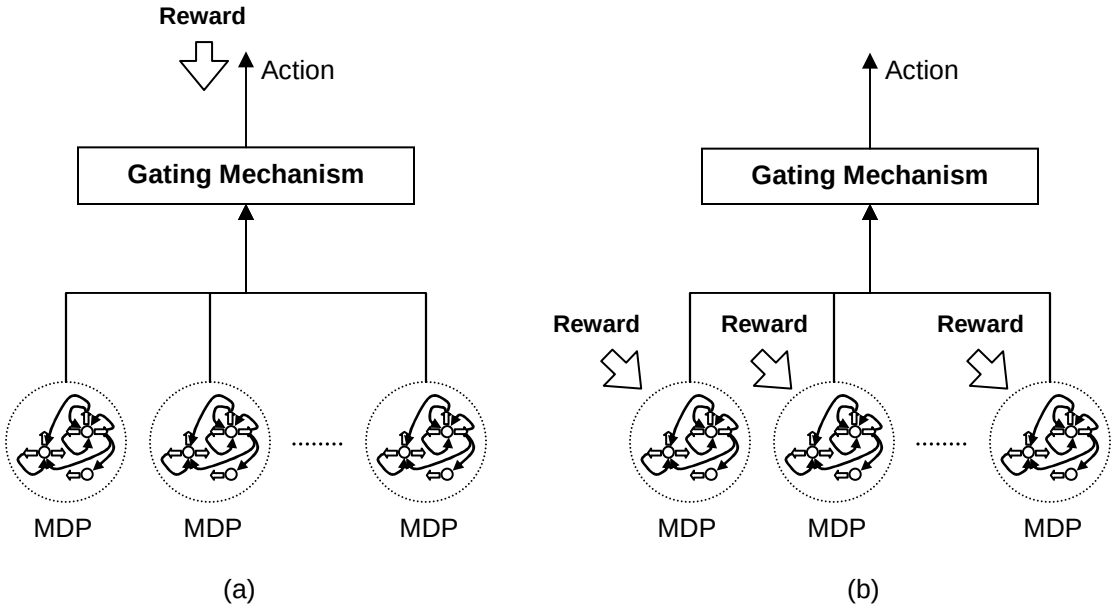


Figure 5. Two possible reward structures for gated MDPs. In the first case (a), the reward is global, and appears after the gating mechanism. In the second case (b), the reward is defined locally for each MDP.

The advantage of the first approach (Figure 5.a) is that in this case the rewards can be constructed around any dialogue performance metric. Optimal behavior is defined differently across various domains and interaction types. In a flight-reservation system, speed and accuracy are important. By contrast, in a tutoring system, these are no longer the primary goals; rather the performance of the system is better measured in terms of how much and how well users can learn. With a post-gating reward structure, global rewards can be constructed around any such metrics, while this is not possible with the second setting (5.b). At the same time, constructing the rewards in this fashion can create a mismatch condition: the rewards reflect the system's performance in terms of both the task and the error handling control, while we are trying to use them to learn only the error handling policies. The problem is further complicated in this case by

the fact that we are facing an atypical, multi-agent reinforcement learning problem (see the following Learning Algorithms subsection).

In the second approach (see Figure 5.b), the rewards are defined locally for each individual MDP, based on the underlying state-space. For instance, for the concept-MDPs the high confidence state can be associated with a high, positive reward. In this approach, we are faced with the standard reinforcement learning problem, and typical model-based learning algorithms can be applied. The disadvantage is that in this setting we cannot target for optimization any given dialogue performance metric. The optimization is now local and targeted only at the error management aspects.

Learning Algorithms

The learning algorithms to be used are closely related to the choice of reward structure described above.

In the post-gating reward setting (Figure 5.a), we are faced with a somewhat atypical reinforcement learning problem. A number of MDPs are running in parallel, and a fixed gating mechanism dictates which one will actually perform an action at each point in time; the others are forced to perform "No Action" (or alternatively, they don't take a time-step). The reward reflects the quality of the concerted actions of these MDPs. The literature search performed so far has not revealed any previous reinforcement learning work precisely addressing this setting. The subject of multiple MDPs acting in collaboration is relatively new in the reinforcement learning community. So far the focus has been on learning the optimal gating function when the policies for the individual MDPs can be relatively easily established [3]. The problem I am faced with is the reverse: given a fixed gating function, learn the best policy for each of the individual MDPs. Recently, Chang et al [14] have proposed an algorithm which allows an agent to learn a near-optimal policy in a similar multi-agent setting. I intend to explore this reinforcement learning problem in detail. A practical, scalable solution would allow the proposed error handling decision process to target any dialogue performance metric for optimization. At the same time, it would represent a valuable contribution to the reinforcement learning community.

In the local rewards setting (Figure 5.b), we are faced with multiple but standard reinforcement learning problems and typical algorithms can be used to derive the optimal policies for each individual MDP. In this case, I anticipate using model-based learning methods, in conjunction with various exploration techniques, such as epsilon-greedy or Boltzmann.

Evaluation

The primary evaluation of the proposed model will address its performance and scalability characteristics.

The performance of the proposed model will be evaluated in terms of a number of objective and subjective metrics such as measures of task completion, efficiency, the number and lengths of error segments, user satisfaction, etc. If a post-gating reward structure is employed, the primary metric for evaluating performance will be the one used to construct the rewards (since the model directly targets that metric for optimization). However, if a local reward structure is used, no explicit primary metric exists, and the performance of the model will be assessed in terms of the various global dialogue metrics proposed above.

The evaluations will be performed in the context of a RavenClaw based spoken dialogue system operating with a sizable, practical task (e.g. RoomLine, CMU Let's Go!! Bus Information System). Initially, a handcrafted policy will be designed and used in exploration mode in a data collection effort (DC-2T) to construct a training corpus. Once the model is trained, a separate

set of experiments (DC-2E) will be performed to evaluate the learned policy. The learned policy will be evaluated against the training policy (i.e. the handcrafted exploratory policy), and a corresponding increase in performance is expected. Two additional baselines can be constructed if time permits[12]: one based on the deterministic version of the initial handcrafted policy, and one based on a version of the system which uses only the most common conversational error handling strategies. These baselines will provide more information about the relative gains of using a reinforcement learning process to optimize the policy versus only using a large number of conversational error handling strategies with a heuristic error handling decision process.

The scalability property of the proposed model will be empirically validated by its deployment in a practical, large scale spoken dialogue system during the learning and evaluation phases described above. Additionally, I intend to perform a theoretical analysis of the scalability, aimed at deriving bounds on the model growth (number of parameters) and data requirements as a function of task (dialogue task tree) size.

Timeline (see Appendix A)

The timeline for completing this proposed work item includes:

- Further investigate the proposed reinforcement learning models. The issues under consideration are: the definition of the gating function heuristic, the structure and initial policies for the concept- and topic-level MDPs, the reward structure, the details of the learning algorithms, and the performance metrics to be used in the evaluation. Implement necessary support for these mechanisms in the RavenClaw dialogue management framework (12 months, timeline item 2);
- Collect data for training the proposed model (3 months, timeline item 9);
- Collect the data for evaluating the proposed model (2 months, timeline item 11);
- Train and evaluate the model (6 months, timeline item 10, overlapped with 10 and 11);
- Write a paper describing the experiments and the error handling decision process (3 months, timeline item 13);

If the initial results contradict the expectations, timeline items 14 and 16 allocate 6 months for further data collection and investigations of alternative models.

If the initial results confirm the expectations, this additional time will be used to perform an in-depth evaluation of the reusability and adaptability of the proposed error handling decision process (see extension work item below).

▷ 6.4.2. EXTENSION WORK ITEM: In-Depth Evaluation of Reusability and Adaptability

The reusability of the learned error handling policies can be directly validated by migrating policies learned in a certain spoken dialogue system to another spoken dialogue system, and analyzing the resulting behavior. An in-depth study of adaptability can also be performed by monitoring the changes in the model parameters and in the system performance throughout an extended period of continuous use. These analyses will require a new data collection effort, using a different RavenClaw-based spoken dialogue system.

[12] These two baselines require additional data collection efforts.

7. Overall Timeline

The anticipated overall timeline for the research program presented in this document is shown in Appendix A. The timeline spans 24 months and subsumes the four proposed work items, the data collection process and a schedule for writing up and publishing the results of the work. Four major milestones, defined roughly every 6 months, will help track progress and ensure that the work is advancing as expected.

Milestone 1 [7 months]: the development of conversational strategies for misunderstandings and non-understandings is completed; a training corpus for the belief-updating model is available; the lower and upper baseline evaluations of the non-understanding strategies are completed;

Milestone 2 [12 months]: the development and evaluation of the belief updating model is completed; the development of relevant indicators for error handling strategies is completed; the development of the error handling strategies is completed;

Milestone 3 [18 months]: the first set of experiments using the proposed reinforcement learning approach is completed;

Milestone 4 [24 months]: thesis defense.

8. References

[1] Ardissono, L., Boella, G., Damiano, R., 1998 – *"A plan-based model of misunderstandings in cooperative dialogue"*, in International Journal of Human-Computer Studies / Knowledge Acquisition, 1998.

[2] Bansal, D., Ravishankar, M.K., 1998 – *"New Features for Confidence Annotation"*, in Proceedings of ICSLP'98

[3] Bererton, C., Gordon, G., Thrun, S., 2003 – *"Auction Mechanism Design for Multi-Robot Coordination"*, under review, available from http://www-2.cs.cmu.edu/~curt/research.html#publications

[4] Bohus, D., Rudnicky, A., 2001 – *"Modeling the Cost of Misunderstandings in the CMU Communicator Spoken Dialogue System"*, in Proceedings of ASRU-2001, Madonna di Campiglio, Italy

[5] Bohus, D., Rudnicky, A., 2002 – *"Integrating Multiple Knowledge Sources for Utterance Level Confidence Annotation in the CMU Communicator Spoken Dialogue System"*, Technical Report CS-190, November 2002, Carnegie Mellon University

[6] Bohus, D., Rudnicky, A., 2002 – *"LARRI: A Language-Based Maintenance and Repair Assistant"*, in Proceedings of IDS-2002, Kloster Irsee, Germany

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

[8] Bousquet-Vernhettes, C., Privat, R., Vigouroux, N., 2003 – *"Error handling in spoken dialogue systems: toward corrective dialogue"*, in Proceedings of Workshop on Error Handling in Spoken Dialogue Systems, Chateau d'Oex-Vaud, Switzerland, 2003.

[9] Bouwman, G., Hulstijn, J., 1998 – *"Dialogue Strategy Redesign with Reliability Measures"*, in Proceedings of the First International on Language Resources and Evaluation, 1998.

[10] Cahn, J.E., Brennan, S.E., 1999 – *"A Psychological Model of Grounding and Repair in Dialog"*, in Working papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems, pp. 25-33, Menlo Park, CA, 1999.

[11] CALO Project Website at: http://www.ai.sri.com/project/CALO

[12] 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, 2001.

[13] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjalmsson, H., Yan, H., 1999 – *"Embodiment in Conversational Interfaces: Rea"*, in Proceedings of CHI'99, pp.520-527, Pittsburgh, PA, 1999

[14] Chang, Y.H., Ho, T., Kaelbling, L.P., 2003 – *"All Learning is Local: Multi-Agent Learning in Global Reward Games"*, in Proceedings of NIPS'2003.

[15] Chase, L., 1997 – *"Error-Responsive Feedback Mechanisms for Speech Recognizers"*, Ph.D. Thesis, CMU, 1997

[16] Clark, H.H., Schaefer, E.F., 1989 – *"Contributing to Discourse"*, in Cognitive Science, vol 13, 1989.

[17] Clark, H.H., Brennan, S.E., 1991 – *"Grounding in Communication"*, in L.B. Resnick, J. Levine and S.D. Teasley, editors, Perspectives on Socially Shared Cognition, 1991.

[18] Cox, S., Rose, R., 1996 – *"Confidence Measures for the SwitchBoard Database"*, in Proceedings of ICASSP'96

[19] Ferguson, G., Allen, J., 1998 – *"TRIPS: An Intelligent Integrated Problem-Solving Assistant"*, in Proceedings of AAAI-98, Madison, WI, July 1998.

[20] Gorin, A.L., Riccardi, G., Wright, J.H., 1997 – *"How May I Help You?"*, Speech Communication, vol 23, pp. 113-127, 1997

[21] Gorrell, G., Lewin, I., Rayner, M., 2002 – *"Adding Intelligent Help to a Mixed Initiative Spoken Dialogue System"*, in Proceedings of ICSLP'02, Denver, CO.

[22] Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D., Wirén M. 2000 – *"AdApt – a multimodal conversational dialogue system in an apartment domain"*, in Proceedings of ICSLP'00, Beijing, China, 2000.

[23] Hazen, T-J., Burianek, T., Polifroni, J., Seneff, S., 2002 – *"Recognition Confidence Scoring for Use in Speech Understanding Systems"*, in Computer Speech and Language, vol. 16, no. 1, pp 49-67, January, 2002.

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

[25] Horvitz, E., Paek, T., 1999 – *"A Computational Architecture for Conversation"*, in Proceedings of the Seventh International Conference on User Modeling, pp. 201-210, Banff, Canada, June 1999.

[26] Horvitz, E., Paek, T., 2001 – *"Harnessing Models of User's Goals to Mediate Clarification Dialog in Spoken Language Systems"*, in Proceedings of the Eighth International Conference on User Modeling, Sonthofen, Germany, July 2001.

[27] Horvitz, E., Paek, T., 2000 – *"DeepListener: Harnessing Expected Utility to Guide Clarification Dialog in Spoken Language Systems"*, in Proceedings of ICSLP'00, Beijing, China, 2000.

[28] Kaelbling, L.P., Littman, M.L., Moore, A.W., 1996 – *"Reinforcement Learning: A Survey"*, in Journal of Artificial Intelligence Research, vol. 4, pp. 237-285, 1996

[29] Kamal, N., Rayid, G., 2000 – *"Analyzing the Effectiveness and Applicability of Co-training"*, in Ninth International Conference on Information and Knowledge Management, pp 86-93, 2000.

[30] Komatani, K., Kawahara, T., 2000 – *"Generating Effective Confirmation and Guidance Using Two-Level Confidence Measures for Dialogue Systems"*, in Proceedings of ICSLP'00.

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

[32] Lemon, O., Gruenstein, A., Cavedon, L., Peters, S., 2002 – *"Multi-Tasking and Collaborative Activities in Dialogue Systems"*, in Proceedings of SIGDIAL-2002, pp. 113-124, Philadelphia, PA, 2002.

[33] Levin, E., Pieraccini, R., Eckert, W., 2000 – *"A Stochastic Model of Human-Machine Interaction for Learning Dialogue Strategies"*, in IEEE Transactions on Speech and Audio Processing, Vol. 8, No. 1, January 2000.

[34] Levow, G-A., 1998 – *"Characterizing and Recognizing Spoken Corrections in Human-Computer Dialogue"*, in Proceedings of COLING'98, pp. 736-742

[35] Litman, D., Hirschberg, J., Swerts, M., 2001 – *"Predicting User Reactions to System Errors"*, in

Proceedings of ACL'01, Toulouse, France, July 2001.

[36] Maes, P., Brooks, R., 1990 – *"Learning to Coordinate Behaviors"*, Proceedings of the 8th National Conference on Artificial Intelligence, pp. 792-802, 1990.

[37] McRoy, S.W., Hirst, G., 1995 – *"The Repair of Speech Act Misunderstandings by Abductive Inference"*, in Computational Linguistics, vol. 21, nr. 4, pp. 435-478, 1995.

[38] McTear, M., O'Neill, I., Hanna, P., Liu, X., 2003 – *"Handling errors and determining confirmation strategies – an object-based approach"*, in Proceedings of Workshop on Error Handling in Spoken Dialogue Systems, Chateau d'Oex-Vaud, Switzerland, 2003.

[39] Mitchell, T., 1999 – *"The Role of Unlabeled Data in Supervised Learning"*, in Proceedings of the Sixth International Colloquium on Cognitive Science, San Sebastian, Spain, 1999.

[40] Niimi, Y., Kobayashi, Y., 1996 – *"A Dialog Control Strategy Based on the Reliability of Speech Recognition"*, in Proceedings of ICSLP'96, pp. 534-537, 1996.

[41] Paek, T., 2001 – *"Empirical Methods for Evaluating Dialog Systems"*, in Proceedings of the 2nd SIGDial Workshop on Discourse and Dialogue, Aalborg, Denmark, September 2001.

[42] Paek, T., Horvitz, E., 2000 – *"Conversation as Action Under Uncertainty"*, in Proceedings of the Sixteenth Conference on Uncertainty and Artificial Intelligence, Stanford, CA, June 2000.

[43] Paek, T., Horvitz, E., 2000 – *"Grounding Criterion: Toward a Formal Theory of Grounding"*, Microsoft Research Technical Report, MSR-TR-2000-40, April 2000.

[44] Raux, A., Eskenazi, M., 2003 – *"Non-native Users in the Let's Go!! Spoken Dialogue System: Dealing with Linguistic Mismatch"*, submitted to HLT/NAACL 2004.

[45] Raux, A., Langner, B., Black, A., Eskenazi, M., 2003 – *"LET'S GO: Improving Spoken Dialogue Systems for the Elderly and Non-natives"*, in Proceedings of Eurospeech-2003, Geneva, Switzerland

[46] Raux, A., 2003 – "EUREKA: Dialogue-based Information Retrieval for Very Low Bandwidth Environments", http://www.cs.cmu.edu/~antoine/11743/

[47] Rickel, J., Lesh, N., Rich, C., Sidner, C., Gertner, A., 2001 – *"Building a Bridge between Intelligent Tutoring and Collaborative Dialogue Systems"*, in Proceedings of the Tenth International Conference on AI in Education, pp 592-594, IOS Press, May 2001.

[48] Rosenfeld, R., Olsen, D., Rudnicky, A., 2000 – *"A Universal Human-Machine Speech Interface"*, Technical Report CMU-CS-00-114, School of Computer Science, Carnegie Mellon University, March 2000.

[49] Rudnicky, A., Thayer, E., Constantinides, P., Tchou, C., Stern, R., Lenzo, K., Xu, W., Oh, A., 1999 – *"Creating natural dialogs in the Carnegie Mellon Communicator System"*, in Proceedings of Eurospeech, 1999, pp 1531-1534

[50] San-Segundo, R., Pellom, B., Ward, W., 2002 – *"Confidence Measure for Dialogue Management in the CU Communicator System"*, in Proceedings of ICASSP 2000.

[51] 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'02, Denver, CO, 2002.

[52] Scheffler, K., 2002 – *"Automatic Design of Spoken Dialogue Systems"*, Ph.D. Thesis, Cambridge, 2002.

[53] Scheffler, K., Young, S., 2001 – *"Corpus-based dialogue simulation for automatic strategy learning and evaluation"*, in Proceedings of NAACL-2001.

[54] Scheffler, K., Young, S., 2002 – *"Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning"*, in Proceedings of HLT-2002.

[55] Seneff, S., Chuu, C., Cyphers, D.S., 2000 – *"Orion: From On-line Interaction to Off-line Delegation"*, in Proceedings of ICSLP'00, Beijing, China, 2000.

[56] Shin, J., Narayanan, S., 2002 – *"Analysis of User Behavior under Error Conditions in Spoken Dialogs"*, in Proceedings of ICSLP'02, Denver, Colorado.

[57] Singh, S., Litman, D., Kearns, M., Walker, M., 2000 – *"Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System"*, in Journal of Artificial Intelligence Research, vol. 16, pp 105-133, 2000.

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

[59] Spoken Dialogue Systems links at: http://www.cs.cmu.edu/~dbohus/SDS/

[60] Sundbald, O., Sundbald, Y., 1998 – *"Olga – a Multimodal Interactive Information Assistant"*, in Proceedings CHI'98.

[61] Swerts, M., Hirschberg, J., Litman, D., 2000 – *"Corrections in spoken dialogue systems"*, in Proceedings of the ICSLP'00, Beijing, China.

[62] TeamTalk - http://crab.speech.cs.cmu.edu/cgi-bin/Wiki11754/wiki.cgi

[63] Tomko, S., 2003 – *"Speech Graffiti: Assessing the User Experience"*, unpublished Master Thesis, Language Technologies Institute, Carnegie Mellon University.

[64] Traum, D., 1998 – *"On Clark and Schaefer's Contribution Model and its Applicability to Human-Computer Collaboration"*, in Proceedings of the COOP'98 Workshop on Use of Clark's Models of Language for the design of Cooperative Systems, May 1998.

[65] Traum, D., 1999 – *"Computational Models of Grounding in Collaborative Systems"*, in Working Papers of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems", pp 124-131, Menlo Park, CA, 1999.

[66] Traum, D., Dillenbourg, P., 1998 – *"Towards a Normative Model of Grounding in Collaboration"*, in Proceedings of ESSLLI'98 Workshop on Mutual Knowledge, Common Ground and Public Information.

[67] Van den Bosch, A., Krahmer, E., Swerts, M., 2001 – *"Detecting problematic turns in human-machine interactions: Rule-induction versus memory-based learning approaches"*, in Proceedings of ACL'01, pp. 499-506,

[68] Walker, M., Hindle, D., Fromer, J., Di Farbrizzio, G., Mestel, C., 1997 – *"Evaluating Competing Agent Strategies for a Voice Email Agent"*, in Proceedings of Eurospeech'97, 1997.

[69] Walker, M., Litman, D., Kamm, C., Abella, A., 1998 – *"Evaluating Spoken Dialogue Systems with PARADISE: Two Case Studies"*, in Computer Speech and Language, 12-3, 1998.

[70] Walker, M., Passonneau, R., Boland, J., 2001 – *"Quantitative and Qualitative Evaluation of the DARPA Communicator Spoken Dialogue Systems"*, in Proceedings of ACL'2001.

[71] Walker, M., Wright, J., Langkilde, I., 2000 – *"Using Natural Language Processing and Discourse Features to Identify Understanding Errors in a Spoken Dialogue System"*, in Proceeding of the 17'th International Conference of Machine Learning, pp 1111-1118, 2000.

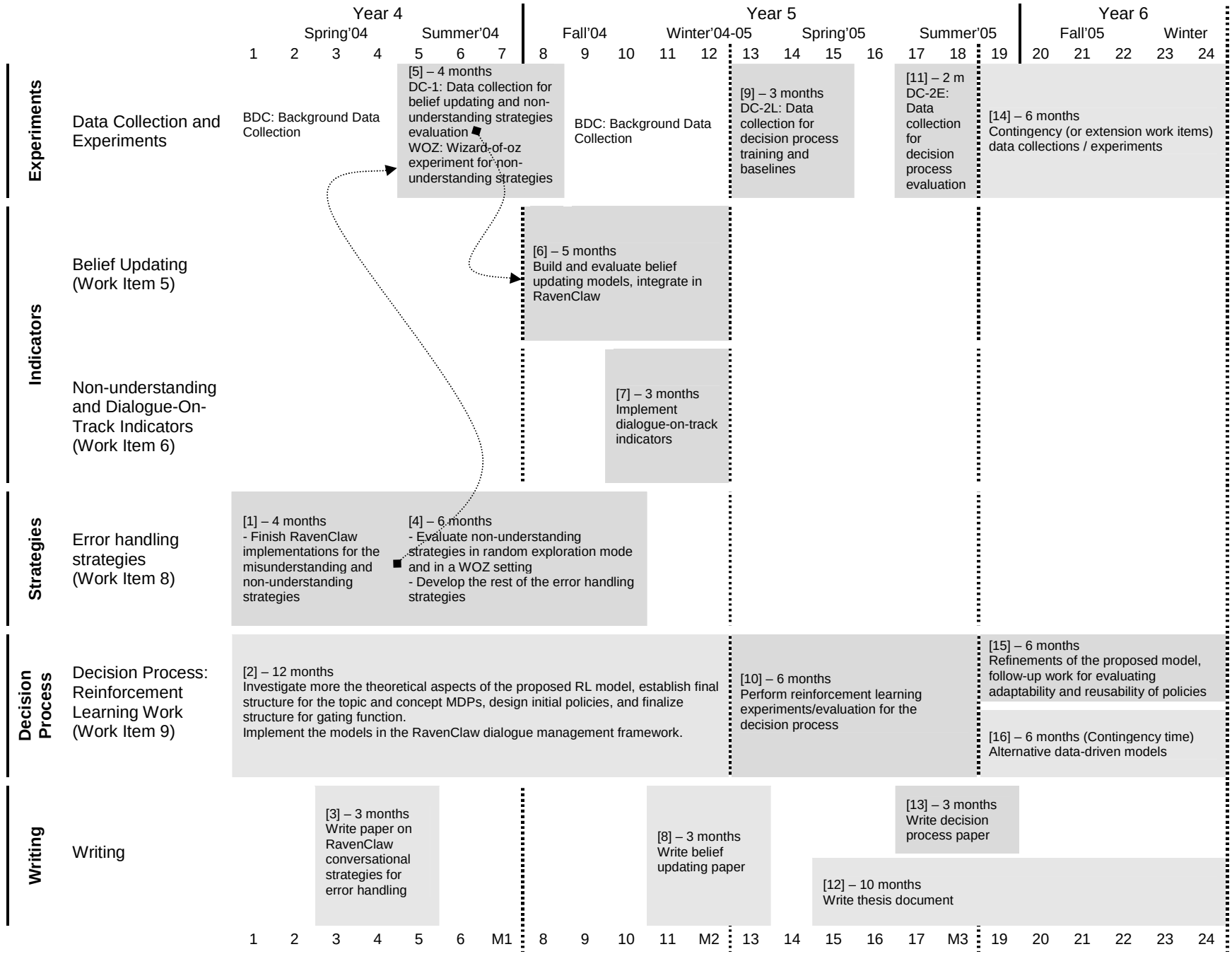
[72] Wang, Y-F.H., Hamerich, S.W., Schless, V., 2003 – *"Multi-Modal and Modality Specific Error Handling in the GEMINI Project"*, in Proceedings of Workshop on Error Handling in Spoken Dialogue Systems, Chateau d'Oex-Vaud, Switzerland, 2003.

[73] Xu, W., Rudnicky, A., 2000 – *"Language Modeling for Dialogue System"*, in Proceedings of ICSLP'00, Beijing, China.

[74] Yankelovich, N., Levow, G.A., Marx, M., 1995 – *"Designing SpeechActs: Issues in Speech User Interfaces"*, in Proceedings of CHI'95, pp. 369-376, 1995.

[75] Zollo, T., 1999 – *"A Study of Human Dialogue Strategies in the Presence of Speech Recognition Errors"*, in Working Notes of AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems, November, 1999.

[76] Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T.J., Hetherington, L., 2000 – *"JUPITER: A Telephone-Based Conversational Interface for Weather Information"*, in IEEE Transactions on Speech and Audio Processing, vol. 8, no. 1, January 2000.


Year 4
Spring'04
Summer'04
Year 5
Fall'04
Winter'04-05
Spring'05
Summer'05
Year 6
Fall'05
Winter
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Experiments
Data Collection and Experiments
BDC: Background Data Collection
[5] – 4 months
DC-1: Data collection for belief updating and non-understanding strategies evaluation
WOZ: Wizard-of-oz experiment for non-understanding strategies
BDC: Background Data Collection
[9] – 3 months
DC-2L: Data collection for decision process training and baselines
[11] – 2 m
DC-2E: Data collection for decision process evaluation
[14] – 6 months
Contingency (or extension work items) data collections / experiments
Indicators
Belief Updating (Work Item 5)
[6] – 5 months
Build and evaluate belief updating models, integrate in RavenClaw
Non-understanding and Dialogue-On-Track Indicators (Work Item 6)
[7] – 3 months
Implement dialogue-on-track indicators
Strategies
Error handling strategies (Work Item 8)
[1] – 4 months
- Finish RavenClaw implementations for the misunderstanding and non-understanding strategies
[4] – 6 months
- Evaluate non-understanding strategies in random exploration mode and in a WOZ setting
- Develop the rest of the error handling strategies
Decision Process
Decision Process: Reinforcement Learning Work (Work Item 9)
[2] – 12 months
Investigate more the theoretical aspects of the proposed RL model, establish final structure for the topic and concept MDPs, design initial policies, and finalize structure for gating function.
Implement the models in the RavenClaw dialogue management framework.
[10] – 6 months
Perform reinforcement learning experiments/evaluation for the decision process
[15] – 6 months
Refinements of the proposed model, follow-up work for evaluating adaptability and reusability of policies
[16] – 6 months (Contingency time)
Alternative data-driven models
Writing
Writing
[3] – 3 months
Write paper on RavenClaw conversational strategies for error handling
[8] – 3 months
Write belief updating paper
[13] – 3 months
Write decision process paper
[12] – 10 months
Write thesis document
1 2 3 4 5 6 M1 8 9 10 11 M2 13 14 15 16 17 M3 19 20 21 22 23 24

Appendix A

Updating beliefs over top-N candidate values for a concept in a dynamic belief net

Problem statement

Given an initial system belief (probability distribution) over the set of candidate values for a concept, and a set of features describing a system action and the follow-up user response, develop a model which accurately updates the initial belief in light of the observed features.

Model

I propose a model based on a dynamic belief network (see Figure 6). The structure of the network will be fixed a priori, and the parameters will be learned from data. The variable of interest, *C*, will model the belief over the candidate concept values. The evidence notes will encode the system action and the features characterizing the decoded user response (see also Section 6.2.2).

For concepts which can have a large number of possible values, such as dates, times, city names, etc, assessing the belief over all the possible values becomes quickly intractable because of the significant increase in the number of model parameters. To address this problem, I propose to track the beliefs only over the top-*N* candidate values for a concept (where *N* is a small integer, like 3 or 5).

Briefly stated, the proposed approach works as follows. A mapping is used in each time-step to encode the relevant information about the top-*N* candidate values for *C* into a code variable *X*. The belief network performs the update over *X*. Finally, the mapping is used in reverse to reconstruct a belief over the top-*N* values for *C*. The proposed encoding will help limit the number of model parameters, and will also facilitate generalization by abstracting over the particularities of each concept (e.g. number of possible values, etc)

Before presenting the proposed abstraction in more detail, I will introduce an example and a couple of notations to facilitate the explanation. Consider the following fictitious interaction (the recognition results are represented in brackets following the actual user response):

```
S:    Where are you flying from?
U:    Austin
      [City = Boston/0.6; Austin/0.35]
S:    Did you say you were flying from Boston?
U:    No, Austin
      [No], [City = Austin/0.8; Aspen/0.1]
```

The initial system belief over the departure city concept is: {Boston/0.6; Austin/0.35}. The system performs an explicit confirmation on Boston, and obtains the result {Austin/0.8; Aspen/0.1}

Let N=3 be the maximum number of candidate values tracked. At each time t, we therefore have a distribution over the top-*N* values for concept *C*, and are interested in updating this distribution for time t+1. In the sequel, by top-i value I will denote the value with the i-th highest probability score in the distribution. Note that the recognition results at each time step can contain new values (different from the ones already in the top-*N*, like "Aspen" in the example above). The model must be able to incorporate these values. Let *M* be the maximum number of new alternatives values contained in a recognition hypothesis that the model can incorporate (*M* will also have a small value like 2 or 3). The abstract variable *X* is then defined to range over the integers from 1 to N+M+1, and it encodes the values for concept *C* as follows:

for i between 1 and N,

i encodes the top-i value at time t. In other words, at time t:

P(X=i) = P(C=top-i value).

In the given example, at time t we have:

P(X=1) = P(C=Boston) = 0.6

P(X=2) = P(C=Austin) = 0.35

for i between 1 and M,

i+N encodes the top-i *new* value contained in the recognition result (only the values which are not already in top-N at time t are considered).

In the given example, assuming N=3, then i=4 will encode "Aspen".

finally, i = N+M+1 encodes "other", i.e. any other value, different from the top-N values and the new top-M values from the recognition.

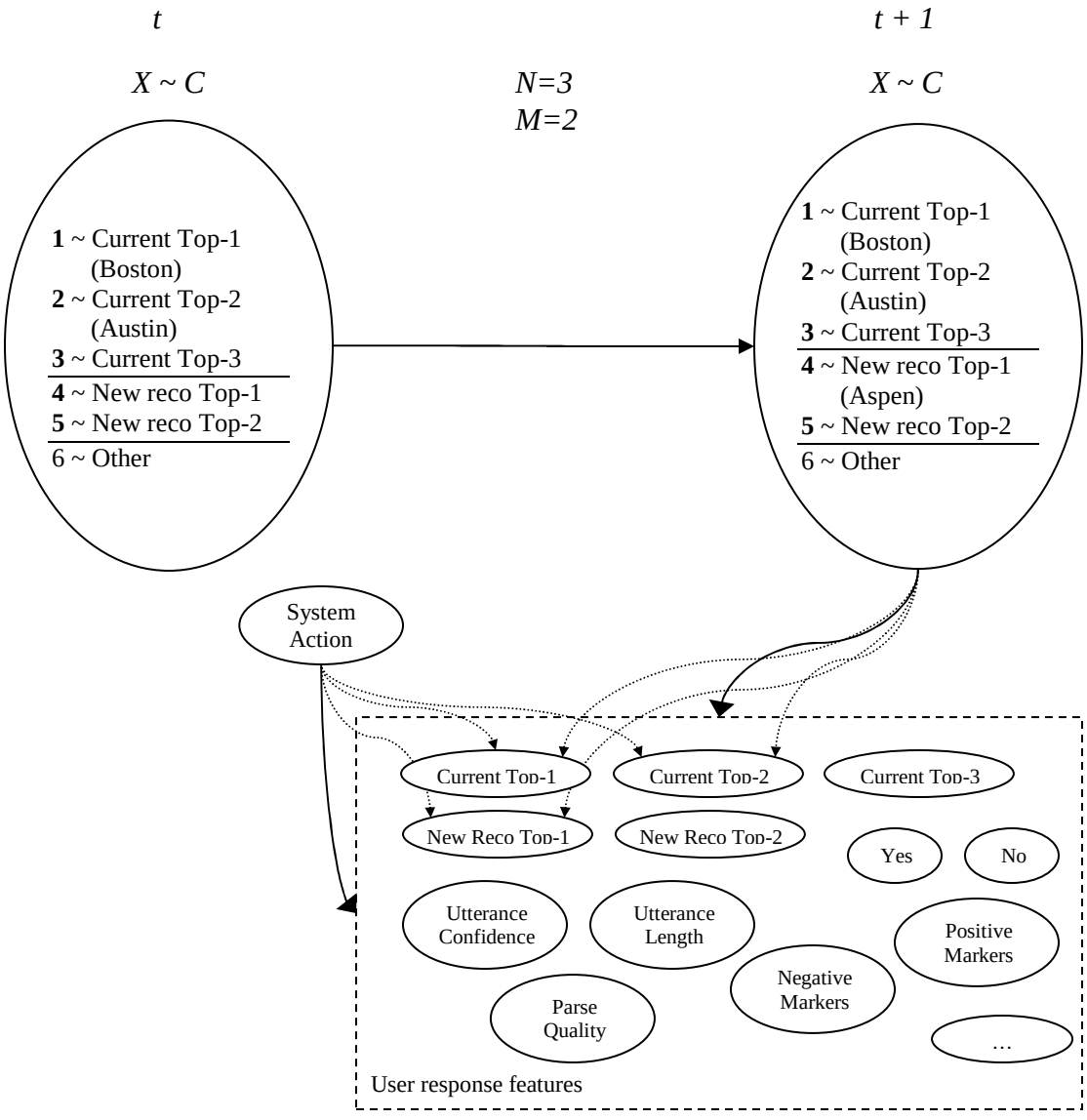


Figure 6. The figure illustrates the structure of the proposed dynamic belief net. The network tracks the belief over the code variable X, which is used to abstract over the top-N candidate values for concept C.

The operation of the model is illustrated in Figure 6. At each time step, the system belief for concept C is known. Following the proposed mapping, we can construct the probabilities P(X=i), for i = 1 .. N. (the rest of the probability mass will be distributed over the remaining values of X: N+1 to N+M+1). The network contains N evidence nodes for the presence of the encoded top-N values in the user response (Current Top-1, etc), as well as M evidence nodes for the presence of a maximum of M other values for the concept in the user response (New Reco Top-1, etc).

The model parameters will be learned from labeled data. Each training instance will contain a current belief for time t, as well as all the features characterizing the user response and system action (these are automatically available). Following transcriptions, the data will be labeled with the true value of the concept, as uttered by the user. This value will be used to construct the optimal target belief at time t+1 (i.e. in the example above, at t+1 during training we have X = 2).

At runtime, inference will be performed to construct the distribution over X at time t+1. Since the top-N values at time t are known, and so are the top-M new recognition values, the mapping can be used in reverse to construct the belief over a maximum of N+M values for the concept C. In the final step, the values are order by probability, and only the new top-N values are retained for the next step (note that because the new top-N values can be different from the old ones, the particular encoding used is potentially different in each time step).