Building a robust dialogue system with limited data

Building a Robust Dialogue System with Limited Data *

Sharon J. Goldwater, Elizabeth Owen Bratt, Jean Mark Gawron, and John Dowding†
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
{*goldwater, owen, gawron, dowding*}*@ai.sri.com*

Abstract

We describe robustness techniques used in the CommandTalk system at the recognition level, the parsing level, and the dialogue level, and how these were influenced by the lack of domain data. We used interviews with subject matter experts (SME's) to develop a single grammar for recognition, understanding, and generation, thus eliminating the need for a robust parser. We broadened the coverage of the recognition grammar by allowing word insertions and deletions, and we implemented clarification and correction subdialogues to increase robustness at the dialogue level. We discuss the applicability of these techniques to other domains.

1 Introduction

Three types of robustness must be considered when designing a dialogue system. First, there is robustness at the recognition level. When plentiful data is available, a robust n-gram language model can be produced, but when data is limited, producing a robust language model for recognition can be problematic. Second, there is robustness at the level of the parser. Robust parsing is often achieved by combining a full parser with a partial parser and fragment-combining rules, but even then some utterances may be correctly recognized, only to be parsed incorrectly or not at all. Finally, there is robustness at the dialogue level. Utterances may be uninterpretable within the context of the dialogue due to errors on the part of either the system or the user, and the dialogue manager should be able to handle such problems gracefully.

Our CommandTalk dialogue system was designed for a highly specialized domain with little available data, so finding ways to build a robust system with limited data was a major concern. In this paper, we discuss our methods and their applicability to other domains. Section 2 gives a brief overview of the CommandTalk system. In Section 3, we discuss the approach we took to building recognition, understanding, and generation models for CommandTalk, and how it relates to the first two types of robustness mentioned. Section 4 discusses additional robustness techniques at the recognizer level, and Section 5 describes dialogue-level robustness techniques. Section 6 discusses the applicability of our methods to other domains.

2 CommandTalk

CommandTalk is a spoken-language interface to the ModSAF (Modular Semi-Automated Forces) battlefield simulator, developed with the goal of allowing military commanders to interact with simulated forces in a manner as similar as possible to the way they would command actual forces. CommandTalk allows the use of ordinary English commands and mouse gestures to

- Create forces and control measures (points and lines)
- Assign missions to forces
- Modify missions during execution
- Control ModSAF system functions, such as the map display
- Get information about the state of the simulation

CommandTalk consists of a number of independent, cooperating agents interacting through SRI's Open Agent Architecture (OAA) (Martin et al., 1998). OAA uses a facilitator agent that plans and coordinates interactions among agents during distributed computation. An introduction to the basic CommandTalk agents can be found in Moore et al. (1997). CommandTalk's dialogue component is described in detail in Stent et al. (1999), and its use of linguistic and situational context is described in Dowding et al. (1999).

* This research was supported by the Defense Advanced Research Projects Agency under Contract N66001-94-C-6046 with the Space and Naval Warfare Systems Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency of the U.S. Government.

† Currently affiliated with GO.com

3 The One-Grammar Approach

In a domain with limited data, the inability to collect a sufficient corpus for training a statistical language model can be a significant problem. For CommandTalk, we did not create a statistical language model. Instead, with information gathered from interviews of subject matter experts (SME's), we developed a handwritten grammar using Gemini (Dowding et al., 1993), a unification-based grammar formalism. We used this unification grammar for both natural language understanding and generation, and, using a grammar compiler we developed, compiled it into a context-free form suitable for the speech recognizer as well.

The effects of this single-grammar approach on the robustness of the CommandTalk system were twofold. On the negative side, we presumably ended up with a recognition language model with less coverage than a statistical model would have had. Our attempts to deal with this are discussed in the next section. On the positive side, we eliminated the usual discrepancy in coverage between the recognizer and the natural language parser. This was advantageous, since no fragment-combining or other parsing robustness techniques were needed.

Our approach had other advantages as well. Any changes we made to the understanding grammar were automatically reflected in the recognition and generation grammars, making additions and modifications efficient. Also, anecdotal evidence suggests that the language used by the system often influences the language used by speakers, so maintaining consistency between the input and output of the system is desirable.

4 Utterance-Level Robustness

It is difficult to write a grammar that is constrained enough to be useful without excluding some reasonable user utterances. To alleviate this problem, we modified the speech recognition grammar and natural language parser to allow certain "close-to-grammar" utterances. Utterances with inserted words, such as *Center on Checkpoint 1 now* or *zoom way out* (where *Center on Checkpoint 1* and *zoom out* are grammatical) were permitted by allowing the recognizer to skip unknown words. We also allowed utterances with deleted words, as long as those words did not contribute to the semantics of the utterance as determined by the Gemini semantic rules constraining logical forms. For example, a user could say, *Set speed, 40 kph* rather than *Set speed to 40 kph*. The idea behind these modifications was to allow utterances with a slightly broader range of wordings than those in the grammar, but with essentially the same meanings.

We began by testing the effects of these modifications on in-grammar utterances, to ensure that they did not significantly decrease recognition performance. We used a small test corpus of approximately 800 utterances read by SRI employees. We collected four measures of performance:

	Non-Robust	Robust
Time, CPURT	0.664	1.05
SRR	2.56%	1.70%
AWER	1.68%	2.94%
SER	10.00%	12.07%

Table 1: In-Grammar Recognition Results

- Recognition time, measured in multiples of CPU real time (CPURT). A recognition time of 1xCPURT means that on our CPU (a Sun Ultra2), recognition took exactly as long as the duration of the utterance.
- Sentence reject rate (SRR). The percentage of sentences that the recognizer rejects.
- Adjusted word error rate (AWER). The percentage of words in non-rejected sentences that are misrecognized.
- Sentence error rate (SER). The percentage of sentences in which some sort of error occurred, either a complete rejection or misrecognized word.

Several parameters affected the results, most notably the numerical penalties assigned for inserting or deleting words, and the pruning threshold of the recognizer. Raising the pruning threshold caused both reject and error rates to go down, but slowed recognition. Lowering the penalties caused rejection rates to go down, but word and sentence error rates to go up, since some sentences which had been rejected were now recognized partially correctly, and some sentences which had been recognized correctly now included some errors. Lowering the penalties also led to slower recognition.

Table 1 shows recognition results for the non-robust and robust versions of the recognition grammar on in-grammar utterances. The pruning threshold is the same for both versions and the insertion and deletion penalties are set to intermediate values. Recognition times for the robust grammar are about 60% slower than those of the control grammar, but still at acceptable levels. Reject and error rates are fairly close for the two grammars. Overall, adding robustness to the recognition grammar did not severely penalize in-grammar recognition performance.

We had very little out-of-grammar data for CommandTalk, and finding subjects in this highly specialized domain would have been difficult and expensive. To test our robustness techniques on out-

of-grammar utterances, we decided to port them to another domain with easily accessible users and data; namely, the ATIS air travel domain. We wrote a small grammar covering part of the ATIS data and compiled it into a recognition grammar using the same techniques as in CommandTalk. Unfortunately, we were unable to carry out any experiments, because the recognition grammar we derived yielded recognition times that were so slow as to be impractical. We discuss these results further in Section 6.

5 Dialogue-Level Robustness

To be considered robust at the dialogue level, a system must be able to deal with situations where an utterance is recognized and parsed, but cannot be interpreted within the current system state or dialogue context. In addition, it must be easy for the user to correct faulty interpretations on the part of the system. Contextual interpretation problems may occur for a variety of reasons, including misrecognitions, incorrect reference resolution, and confusion or incompleteness on the part of the user.

The CommandTalk dialogue manager maintains a stack to keep track of the current discourse context and uses small finite-state machines to represent different types of subdialogues. Below we illustrate some types of subdialogues and other techniques which provide robustness at the dialogue level. Note that for each utterance, we write what the system recognizes, not what the user actually says.

5.1 Correction Subdialogues

Ex. 1:

U	1	Create a CEV at 76 53
S	2	⊙
U	3	Put Objective Golf here <*click*>
S	4	⊙ I will locate Objective Golf at FQ 658 583
U	5	I said Objective Alpha
S	6	⊙ I will locate Objective Alpha at FQ 658 853

Allowing the user to correct full or partial utterances can remedy interpretation problems caused by misrecognitions, incorrect reference resolution, or user error.

In Example 1, the system responds to the user's first utterance by producing a rising tone, illustrated by the ⊙ symbol, to indicate successful interpretation and execution of the command, in this case creation of a CEV, a type of vehicle. (Unsuccessful interpretation is indicated by a falling tone, illustrated by the ⊗ symbol.) In utterances 3 through 6, a misrecognition causes the system to perform the wrong behavior. The user initiates a correction subdialogue, and the system goes on to correctly reinterpret the full utterance.

5.2 Implicit Confirmation

Ex. 2:

U	7	CEV, advance to Checkpoint 1
S	8	⊙ 100A11 will proceed to Checkpoint 1

Use of implicit confirmation in combination with correction subdialogues makes it easy to correct faulty interpretations as soon as possible by alerting the user to possible sources of error.

In utterances 7 and 8, the system must resolve the user's reference, "CEV", to a particular unit. It therefore echoes the user's command using the CEV's unique call sign. This makes explicit the system's interpretation of the user's utterance, giving the user a chance to correct the system if necessary.

Note that utterance 4 also contains an implicit confirmation, since the system has resolved the user's gesture to a set of coordinates.

5.3 Clarification Subdialogues

Ex. 3:

U	9	Create a CEV at 72 69
S	10	⊙
U	11	CEV, conduct crater breach facing south
S	12	⊗ There are two CEV's. Do you mean 100A11 or 100A12?
U	13	A12
S	14	⊗ Which point should 100A12 conduct crater breach oriented south at?
U	15	Checkpoint 2
S	16	⊗ There is no Checkpoint 2. Which point should 100A12 conduct crater breach oriented south at?
U	17	I meant Checkpoint 1
S	18	⊙ 100A12 will conduct crater breach oriented south at Checkpoint 1

Clarification subdialogues are generally initiated by the system as a result of errors or incomplete commands on the part of the user.

Example 3 illustrates three different types of problems that can be corrected by system questions. First, the user's reference to "CEV" in utterance 11 is ambiguous, so the system asks a question to determine which CEV the user is referring to. Next, the system asks the user to supply a missing piece of information that is required to carry out the command. Finally, when the user makes an error by referring to a point that doesn't exist, the system prompts for a correction.

6 Discussion and Conclusions

CommandTalk is an example of a successful and robust dialogue system in a domain with limited ac-

cess to both data and subjects. The pre-dialogue version of CommandTalk was used in the STOW (Synthetic Theater of War) '97 ACTD (Advanced Concept Technology Demonstration) exercise, an intensive 48-hour continuous military simulation by all four U.S. military services, and received high praise. The dialogue portion of the system has increased CommandTalk's usefulness and robustness. Nevertheless, several questions remain, not the least of which is whether the robustness techniques used for CommandTalk can be successfully transferred to other domains.

We have no doubt that our methods for adding robustness at the dialogue level can and should be implemented in other domains, but this is not as clear for our parsing and recognition robustness methods.

The one-grammar approach is key to our eliminating the necessity for robust parsing, renders a large corpus for generating a recognition model unnecessary, and has other advantages as well. Yet our experience in the ATIS domain suggests that further research into this approach is needed. Our ATIS grammar is based on a grammar of general English and has a very different structure from that of CommandTalk's semantic grammar, but we were unable to isolate the factor or factors responsible for its poor recognition performance. Recent research (Rayner et al., 2000) suggests that it may be possible to compile a useful recognition model from a general English unification grammar if the grammar is constructed carefully and a few compromises are made. We also believe that using an appropriate grammar approximation algorithm to reduce the complexity of the recognition model may prove fruitful. This would reintroduce some discrepancy between the recognition and understanding language models, but maintain the other advantages of the one-grammar approach.

In either case, the effectiveness of our recognition robustness techniques remains an open question. We know they have no significant negative impact on in-grammar recognition, but whether they are helpful in recognizing and, more importantly, interpreting out-of-grammar utterances is unknown. We have been unable to evaluate them so far in the CommandTalk or any other domain, although we hope to do so in the future.

Another possible solution to the problem of producing a workable robust recognition grammar would return to a statistical approach rather than using word insertions and deletions. Stolcke and Segal (1994) describe a method for combining a context-free grammar with an n-gram model generated from a small corpus of a few hundred utterances to create a more accurate n-gram model. This method would provide a robust recognition model based on the context-free grammar compiled from our unification grammar. We would still have to write only one grammar for the system, it would still influence the recognition model, and we could still be sure that the system would never say anything it couldn't recognize. This approach would require using robust parsing methods, but might be the best solution for other domains if compiling a practical recognition grammar proves too difficult.

Despite the success of the CommandTalk system, it is clear that more investigation is called for to determine how best to develop dialogue systems in domains with limited data. Researchers must determine which types of unification grammars can be compiled into practical recognition grammars using existing technology, whether grammar approximations or other techniques can produce good results for a broader range of grammars, whether allowing word insertions and deletions is an effective robustness technique, or whether we should use other methods altogether.

References

J. Dowding, J. Gawron, D. Appelt, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A Natural Language System for Spoken Language Understanding. In *Proceedings of the Thirty-First Annual Meeting of the ACL*, Columbus, OH. Association for Computational Linguistics.

J. Dowding, E. Owen Bratt, and S. Goldwater. 1999. Interpreting Language in Context in CommandTalk. In *Communicative Agents: The Use of Natural Language in Embodied Systems*, pages 63–67.

D. Martin, A. Cheyer, and D. Moran. 1998. Building Distributed Software Systems with the Open Agent Architecture. In *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, Blackpool, Lancashire, UK. The Practical Application Company Ltd.

R. Moore, J. Dowding, H. Bratt, J. Gawron, Y. Gorfu, and A. Cheyer. 1997. CommandTalk: A Spoken-Language Interface for Battlefield Simulations. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 1–7, Washington, DC. Association for Computational Linguistics.

M. Rayner, B. A. Hockey, F. James, E. Owen Bratt, S. Goldwater, and J. M. Gawron. 2000. Compiling Language Models from a Linquistically Motivated Unification Grammar. Submitted to COLING '00.

A. Stent, J. Dowding, J. Gawron, E. Owen Bratt, and R. Moore. 1999. The CommandTalk Spoken Dialogue System. In *Proceedings of the 37th Annual Meeting of the ACL*. Association of Computational Linguistics.

A. Stolcke and J. Segal. 1994. Precise N-Gram Probabilities from Stochastic Context-free Grammar. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 74–79.