SPEECH REPAIRS: A PARSING PERSPECTIVE

Mark G. Core and Lenhart K. Schubert
mcore,schubert@cs.rochester.edu
http://www.cs.rochester.edu/u/mcore
Computer Science Department
University of Rochester
Rochester NY 14627

ABSTRACT

This paper presents a grammatical and processing framework for handling speech repairs. The proposed framework has proved adequate for a collection of human-human task-oriented dialogs, both in a full manual examination of the corpus, and in tests with a parser capable of parsing some of that corpus. This parser can also correct a pre-parser speech repair identifier producing increases in recall varying from 2% to 4.8%.

1. MOTIVATION

In the discussion below, we adopt the convention of using the term speech repair to include hesitations. Many speech repairs have associated editing terms (*I mean*, *um*), and abridged repairs [6]) consist solely of editing terms (i.e. they have no corrections).

Speech-based dialog systems often attempt to identify speech repairs in the speech recognition phase (prior to parsing) so that speech repairs will not disrupt the speech recognizer's language model ([6],[7],[8]). In such a system, it is then tempting to remove conjectured reparanda (corrected material) and editing terms from the input prior to further processing. There are two issues that need to be addressed in such an approach, one pertaining to dialog interpretation and the other to parsing. First, how can the dialog manager of the system access and interpret these editing terms and reparanda, if the need arises? Such a situation could occur in an example such as *take the oranges to Elmira, um, I mean, take them to Corning*; here reference resolution requires processing of the reparandum. Also, the system might want to access the reparanda and editing terms to see the speaker's original thoughts and any hesitations, for instance as indicators of uncertainty. For more details see [3]. Second, if speech repair identification occurs before parsing, should the parser be made aware of reparanda?

We believe that the answer to the second question should be yes. The parser has more information about the possible grammatical structures in the input than a pre-parser repair identifier and can possibly correct errors made by it. This point applies not only to each speaker's contributions in isolation but also to the interactions between contributions. An example is provided by utterances 11-15 of TRAINS dialog [5] d91-6.1 (Figure 1) where the interleaving of speaker contributions can help identify repairs.

(To fit the example on one line, we have abbreviated the initial part of *u*'s contribution, *move the engine at Avon engine E to*, to *move engine E to*.) Repair detection and correction typically act on only one speaker's stream of words at a time. If for some reason, the corrections *E one*, *en-*, and *engine E one* were not recognized by a pre-parser repair detector, the parser's knowledge of *s*'s correction might help find these repairs. If the dialog parser treats the words of the two speakers as a single stream of data (as ours in fact does), *s*'s correction appears right after the phrase it corrects.

This paper presents a framework that addresses both sorts of issues above. The framework allows for complete phrase structure representations of utterances containing repairs without removing reparanda from the input. Thus structural analyses of repairs are made available to the dialog manager. The idea is to create two or more interpretations for each repair; one interpretation for the corrected utterance and one possibly partial interpretation for what the speaker started to say. Editing terms are considered separate utterances embedded in the main utterance.

The focus of this paper is on the second issue, i.e., testing the ability of a parser to improve pre-parser speech repair identification. We show that by applying the parser's knowledge of grammar and of the syntactic structure of the input to the hypotheses made by the pre-parser repair identifier, we can improve upon those hypotheses.

2. HOW THE PARSER ACCOMODATES REPAIRS

The parser deals with reparanda and editing terms via metarules. The term metarule is used because these rules act not on words but on grammatical structures. Consider the *editing term metarule*. When an editing term is seen[1], the metarule extends copies of all phrase hypotheses ending at the editing term over that term to allow utterances to be formed around it. This metarule (and our other metarules) can be viewed declaratively as specifying allowable patterns of phrase breakage and interleaving [2]. This notion is different from the traditional linguistic conception of metarules as rules for generating new PSRs from given PSRs.[2] Procedurally, we can think of metarules as creating new (discontinuous) pathways for the parser's traversal of the input, and this view is readily implementable.

The repair metarule, when given the hypothetical start and end of a reparandum (say from a language model such as [6]),

```
U: move engine E to            E one    en- engine E one to Bath
S:                engine E one     okay
```

Figure 1: Utterances 11-15 of TRAINS dialog d91-6.1

extends copies of phrase hypotheses over the reparandum allowing the corrected utterance to be formed. In case the source of the reparandum information gave a false alarm, the alternative of not skipping the reparandum is still available.

For each utterance in the input, the parser needs to find an interpretation that starts at the first word of the input and ends at the last word. This interpretation may have been produced by one or more applications of the repair metarule allowing the interpretation to exclude one or more reparanda. For each reparandum skipped, the parser needs to find an interpretation of what the user started to say. In some cases, what the user started to say is a complete constituent: *take the oranges I mean take the bananas*. Otherwise, the parser needs to look for an incomplete interpretation ending at the reparandum end. Typically, there will be many such interpretations; the parser searches for the longest interpretations and then ranks them based on their category: UTT > S > VP > PP, and so on. The incomplete interpretation may not extend all the way to the start of the utterance in which case the process of searching for incomplete interpretations is repeated. Of course the search process is restricted by the first incomplete constituent. If, for example, an incomplete PP were found then any additional incomplete constituent would have to expect a PP.

Figure 2 shows an example of this process on utterance 62 from TRAINS dialog d92a-1.2. Assuming perfect speech repair identification, the repair metarule will be fired from position 0 to position 5 meaning the parser needs to find an interpretation starting at position 5 and ending at the last position in the input. This interpretation (the corrected utterance) is shown under the words in figure 2 The parser then needs to find an interpretation of what the speaker started to say. There are no complete constituents ending at position 5. The parser instead finds the incomplete constituent `ADVBL -> adv • ADVBL`. Our implementation is a chart parser and accordingly incomplete constituents are represented as arcs. This arc only covers the word *through* so another arc needs to be found. The arc `S -> S • ADVBL` expects an `ADVBL` and covers the rest of the input, completing the interpretation of what the user started to say (as shown on the top of figure 2). The editing terms are treated as separate utterances via the editing term metarule. For more details including a discussion of second speaker interruptions see [2],[4]

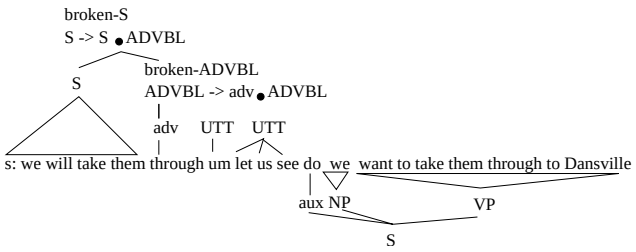

Figure 2: Utterance 62 of d92a-1.2

3. RESCORING A PRE-PARSER SPEECH REPAIR IDENTIFIER

Given that the parser can accept input containing reparanda and editing terms, a pre-parser repair identifier does not have to "clean up" input by removing hypothesized reparanda and editing terms. It can instead give the parser its n-best hypotheses about possible reparanda and editing terms. For this paper, we put aside the question of how the parser determines when utterances end. In the experiments below, the parser will always be given utterance endpoints. For each utterance, the parser can try various hypotheses from the repair identifier. Based on the grammaticality of these hypotheses and any scores previously assigned to them, the parser decides which one is correct.

To test whether such post-correction would improve recall, the parser described in section 2 was connected to Heeman's speech repair identifier [6]. The latter produced up to 100 hypotheses about the speech repairs, boundary tones, and parts of speech associated with the words of each turn in the test corpus. Each hypothesis was given an estimated probability.

Both the parser and Heeman's speech repair identifier were developed and tested on the TRAINS corpus [5]. However, Heeman's testing data was broken into two streams for the two speakers while the test data for the parser merged the two speakers' words into one data stream. The differences in segmentation resulted in different speech repair annotations.

3.1 Experiment One

The first experiment used the parser's speech repair annotations. The version of Heeman's module used is prior to the one reported in [6]. Correspondingly, the recall and precision of this module are lower than current versions. The recall and precision of the model on the test corpus is shown in table 1. The test corpus consisted of 541 repairs, 3797 utterances, and 20,069 words.[3]

To correct Heeman's output, the parser starts by trying his module's first choice. If this results in an interpretation covering the input, that choice is selected as the correct answer. Otherwise the process is repeated with the module's next choice. If all the choices are exhausted and no interpretations are found, then the first choice is selected as correct. This approach is similar to an experiment in [1] except that Bear et al. were more interested in reducing false alarms. Thus, if a sentence parsed without the repair then it was ruled a false alarm. Here the goal is to increase recall by trying lower probability alternatives when no parse can be found.

Repairs correctly guessed	271
False alarms	215
Missed	270
Recall	50.09%
Precision	55.76%

Table 1: Heeman's Speech Repair Results from Exp 1

Repairs correctly guessed	284
False alarms	371
Missed	257
Recall	52.50%
Precision	43.36%

Table 2: Augmented Speech Repair Results from Exp 1

The results of such an approach on the test corpus are listed in table 2. Recall increased by 4.8% (13 cases out of 541 repairs), showing promise in the technique of rescoring the output of a pre-parser speech repair identifier.

One factor relating directly to the effectiveness of the parser at correcting speech repair identification is the percent of fluent or corrected utterances that the parser's grammar covers. In a random sample of 100 utterances from the corpus, 65 received

some interpretation. However, 37 of these utterances are one word long (*okay, yeah*, etc.) and 5 utterances were question answers (*two hours, in Elmira*); thus on interesting utterances, likely to have repairs, accuracy is 39.7%. What happens when a fluent or corrected utterance cannot be parsed is that the parser may pick a low scoring repair hypothesis that eliminates the unparsable material (this may be most of the utterance). This situation results in a false alarm and actual repairs in the input may be missed.

A question raised by this experiment was the effect of knowing utterance boundaries in choosing a repair hypothesis. All of Heeman's repair hypotheses were truncated to fit within utterance boundaries. However, this may have resulted in obviously incorrect hypotheses that the parser could easily eliminate. If Heeman's module had known of utterance boundaries at the outset it could have eliminated these possibilities itself. The baseline measures of the second experiment were adjusted to control for this advantage.

3.2 Experiment Two

In the second experiment, the most recent version of Heeman's repair identifier was used; a baseline measure considering the effect of utterance boundaries was calculated; and Heeman's segmentation of the TRAINS corpus was used. Heeman's segmentation broke the input into two parts, one for each speaker, and further divided those into turns. The author broke turns into utterances as defined by the parser's grammar. Heeman's scoring module worked on a per-turn basis, meaning if a turn had several utterances the parser was not allowed to pick one hypothesis for the first utterance and a different one for the second. The parser scored the different hypotheses based on the number of words that parsed for each hypothesis. So if one hypothesis allowed two utterances to parse, one containing 5 words and another containing 7 words, its score would be 12. The hypothesis with the highest score was picked. In the case of ties, the hypothesis with the higher probability (as assigned by Heeman) was chosen.

To construct a baseline measurement taking into account the effect of utterance boundaries, hypotheses output by Heeman's module that crossed utterance boundaries were eliminated. The top scoring hypothesis out of those remaining was selected for each turn. The resulting recall and precision are shown in table 3. The test corpus for this experiment includes one additional dialog (d93-10.5) giving a total of 20,213 words. The additional dialog and different segmentation and repair annotations result in a corpus of 2295 turns, 3953 utterances and 695 speech repairs. Involving the parser as described above produces the results shown in table 4. Recall increases by 2% (9 repairs out of 695). Actually, there are 30 cases where the parser corrected the output of Heeman's module, but there are also 21 cases where the parser incorrectly rejected Heeman's first choice creating a false alarm and causing a repair to be missed. These instances occurred when the parser's grammar did not recognize the corrected utterance.

Because three aspects of the experiment were changed between experiments one and two, it is difficult to say whether 2% is a more valid measure of increase in recall than the 4.8% measured in experiment one. As a preliminary test, we measured the parsability of 60 turns randomly drawn from this corpus and containing 100 utterances. 63.3% of the turns parsed but if we do not consider turns consisting of one-word utterances and phrasal question answers then only 31.3% of these non-trivial turns parsed. Since experiment one was utterance-based and had a parsing rate of 39.7% on non-trivial utterances, the change in segmentation could have affected the recall rate. Clearly more experiments need to be run to get the correct figure.

Repairs correctly guessed	445
False alarms	125
Missed	250
Recall	64.03%
Precision	78.07%

Table 3 Heeman's Speech Repair Results from Exp 2

Repairs correctly guessed	454
False alarms	749
Missed	241
Recall	65.32%
Precision	37.74%

Table 4: Augmented Speech Repair Results from Exp 2

3.3 Discussion

The first question to be answered about these results is how to address the drop in precision. Up to this point the probabilities assigned by Heeman's module were only used to break ties. Combining these probabilities with the percentage of words parsed and using this score to rank hypotheses could offset the effect of lower probability hypotheses that remove unparsable but fluent material from the input.

A wider coverage grammar would also help, but the parser would still be judging repairs solely on whether they occur in the interpretation constructed by the parser. In addition to grammatical disruption, the parser could also measure syntactic parallelism between a potential reparandum and its correction. This ability needs to be investigated in further detail. Phrase-level parallelism will not likely be enough. An informal search of the test corpus revealed that only 11% of repairs were corrections of complete phrases or clauses. One could modify a statistical parser to return the most likely incomplete and complete constituents at every position in the input. Having incomplete constituents for comparison might allow a useful syntactic parallelism score to be constructed. Or perhaps the role of the parser should merely be to decide whether a particular repair hypothesis fits in the most highly probable parse of the input.

The results of these experiments are promising. Even with low grammatical coverage the parser was able to increase the recall. The remaining missing examples were not recovered either because the parser's grammar did not cover the corrected utterance or Heeman's repair module did not include the repair. Post-hoc analysis is needed to determine whether the majority of errors were the result of the parser or whether we also need to consider how to find repairs not posited by a module such as Heeman's.

In the case of grammar failure, the parser cannot interpret the utterance even if the correct repair hypothesis was chosen. An experiment described in [4] measured utterance parsing accuracy on a corpus of 495 repairs from the TRAINS dialogs. Even though the parser was given perfect speech repair information, only 144 of the 495 repairs appeared in utterances having a complete parse. Thus, the 9 additional repairs (out of 695) found in experiment 2 and the 13 additional repairs (out of 541) in experiment 1 should be considered in light of the fact that

these repairs are in utterances that parse whereas even if the other repairs in these corpora were corrected they might not parse. So the effect of the 9 and 13 repairs on the comprehensibility of the corpora is somewhat greater than the 2% and 4.8% increases in repair recall measured above.

4. CONCLUSION

The dialog parsing framework and implementation presented in this paper show how to extend standard parsers to handle speech repairs. Such an approach allows the parser's knowledge of the possible grammatical structures of the input to impact speech repair identification, resulting in increases in recall varying from 2% to 4.8%. This approach also provides a structural representation of reparanda enabling a dialog system to track the speaker's "train of thought" (or as mentioned, to support reference resolution).

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation grants IRI-9503312 and 5-28789. Thanks to James Allen and Amon Seagull for their help and comments on this work. Thanks to Peter Heeman for providing data and guidance for the paper.

NOTES

1. The parser's lexicon has a list of 35 editing terms that activate the editing term metarule.
2. For instance, a traditional way to accommodate editing terms might be via a metarule,
X -> Y Z ==>$ X -> Y editing-term Z, where X varies over categories and Y and Z vary over sequences of categories. However, this would produce phrases containing editing terms as constituents, whereas in our approach editing terms are separate utterances.
3. Specifically the dialogs used were d92-1 through d92a-5.2;
d93-10.1 through d93-10.4; and d93-11.1 through d93-14.2. The language model was never simultaneously trained and tested on the same data.

REFERENCES

[1] Bear, J., Dowding, J., and Shriberg, E. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL 92)*, 56-63.

[2] Core, M. and Schubert L. 1998. Implementing parser metarules that handle speech repairs and other disruptions. In Cook. D. (ed.), *Proceedings of the 11th International FLAIRS Conference*. Sanibel Island.

[3] Core, M. and Schubert L. 1999. A model of speech repairs and other disruptions. Working notes of the *AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*. Cape Cod.

[4] Core, M. and Schubert L. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 99).* College Park.

[5] Heeman, P.A. and Allen, J. F. 1995. the TRAINS 93 dialogues. TRAINS Technical Note 94-2, Department of Computer Science, University of Rochester, Rochester NY 14627-0226.

[6] Heeman, P. A. and Allen, J. F. 1997. Intonational boundaries, speech repairs, and discourse markers: modeling spoken dialog. *In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 97).* Madrid, 254-261.

[7] Siu, M.-h. and Ostendorf, M. 1996. Modeling disfluencies in conversational speech. In *Proceedings of the 4rd International Conference on Spoken Language Processing (ICSLP-96)*, 386-389.

[8] Stolcke, A. and Shriberg, E. 1996. Statistical language modeling for speech disfluencies. In *Proceedings of the International Conference on Audio, Speech, and Signal Processing (ICASSP).*