AN IMPROVED MODEL FOR RECOGNIZING DISFLUENCIES IN CONVERSATIONAL SPEECH

Mark Johnson, Eugene Charniak and Matthew Lease

Brown Laboratory for Linguistic Information Processing
Brown University

ABSTRACT

This paper presents a novel metadata extraction (MDE) system for automatically detecting edited words, fillers, and self-interruption points in conversational speech. Our edit word detection sub-system combines a Tree Adjoining Grammar (TAG) noisy channel model, a statistical syntactic language model, and a MaxEnt reranker. Hand-built, deterministic rules are used to detect fillers. Self-interruption points are explicitly determined by detected fillers and edited words. We have evaluated our system for these three tasks on two types of input: manually annotated words and automatically recognized speech-to-text tokens. In all six cases, our system has improved the state-of-the-art, as measured in a recent blind evaluation.

1. INTRODUCTION

In previous work [1, 2], we presented a noisy channel model for detecting speech repairs, and showed that this performs better than a simple word-by-word classifier approach. This paper extends these methods to three specific metadata extraction (MDE) tasks:

Edit Word Detection (EWD): on a word-by-word basis, discriminate between those words which are and are not part of the reparandum region of a speech repair

Filler Word Detection (FWD): on a word-by-word basis, discriminate between those words which are and are not part of a filler phrase (e.g. "um", "you know", "so anyway", etc.). Furthmore, the type of filler must also be specified: Filled Pause (FP), Discourse Marker (DM), or Explicit Editing Term (EET). FPs are words such as "um", "ah", "eh", etc. DMs signal the speaker's intent to mark a boundary in discourse, and EETs consist of an overt statement from the speaker recognizing the existence of a disfluency [3]

Interruption Point Detection (IPD): for each inter-word gap, predict whether speech becomes disfluent at that point (the previous word ended a reparandum or the next word begins a filler).

In all cases, our goal is to recover metadata annotations consistent with those specified by LDC [3].

Our overall system architecture for performing these tasks is sketched in Figure 1. The noisy channel model we proposed previously [1] formed the basis of the system presented here. As explained in section 2, that model produces 25 different sentence-level hypotheses about the locations of edited words for each sentence it analyzes and scored each analysis using a parsing based language model, as described in section 3. The system takes transcript words segmented into sentences as input. Whereas the earlier noisy channel model simply returns the most probable sentence-level hypothesis, the system described here uses a MaxEnt classifier to rerank these hypotheses and select the highest scoring one [4]. As explained in section 4, an advantage of MaxEnt reranking is that it permits us to use a wide range of additional information to identify the best sentence-level analysis.

Our base system, as described previously [1], detects the components of speech repairs as identified by Shriberg [5], i.e., it identifies a reparandum, an interregnum (possibly empty) and a repair substring for each speech repair. This means that we can use it to detect edited words, but it does not supply enough information for detecting fillers or self-interruption points (IPs). With regard to IPs, our base system only predicts IPs following edits, not those which precede fillers. We have therefore augmented our base system with a set of hand-written deterministic rules for the FWD task (see section 5), and the output of our EWD and FWD sub-systems explictly determines detected IPs.

In section 7, we present experimental results of our system as measured in the recent Fall 2004 Rich Text (RT-04F) MDE blind evaluation [6]. As part of this evaluation, sys-

This work was supported by NSF grants LIS 9720368 and IIS0095940. We would like to thank the LDC, NIST and DARPA for supplying us with the data used in this evaluation, and the RT-04F teams from SRI, ICSI and the University of Washington for the speech recognizer output, detected SU boundaries and IP probabilities that enabled us to take part in the evaluation. In particular, we would like to thank Mari Ostendorf, Dustin Hillard, Yang Liu, Jeremy Kahn, and Jon Fiscus for their tremendous support.

tem performance was assessed on both manually annotated words and automatically recognized speech-to-text tokens. Our system was the the top performer in all six cases.

2. THE TAG CHANNEL MODEL

We have described the TAG channel model is detail previously [1], so we only summarize it here. Following Shriberg's analysis of speech repairs [5] the channel model analyzes speech repairs in terms of the reparandum, interregnum and repair components, as depicted in Figure 2. Only the detected reparandum and interregnum are annotated in its output, since later processing does not depend on the repair words. Specifically, it tags every word in its input string with one of the following tags:

`E]` final word in reparandum,

`E` nonfinal word in reparandum,

`I]` final word in interregnum,

`I` nonfinal word in interregnum, and

_ word not in reparandum or interregnum.

Thus a single analysis produced by the channel model might be something like:

```
a _ flight _ to E Boston E] uh I uh I]
to _ Denver _
```

The chief difficulty in detecting such structures is that they involve crossing dependencies, which lie outside of the expressive power of finite state and context-free grammars. However, crossing dependencies of this kind can be generated by Tree Adjoining Grammars (TAGs). Unlike most TAGs, which are used to describe head-complement and other linguistically important dependencies, the TAGs used here capture the "rough copy" relationship between the reparandum and repair strings. In our system, conventional linguistic dependencies are captured by the syntactic parser based language model described in section 3.

Informally, the TAG transducer in the channel model is responsible for generating the words in the reparandum and the interregnum of a speech repair from corresponding repair. The channel model non-deterministically predicts repairs at every position in the input string, conditioned on the preceding word. Because it is conditioned on the preceding word, it captures the well-known effect that repairs are more likely to occur at the beginning of a sentence than elsewhere. The interregnum is generated by a simple unigram model over words or pairs of words such as *I mean*, etc. Each reparandum word is generated conditioned on the word preceding the reparandum and the repair string that corresponds to the reparandum. The TAG transducer is effectively a simple first-order Markov model (albeit one that captures string non-local dependencies) generating each word in the reparandum conditioned on the preceding word in the reparandum and the corresponding word in the repair.

In our noisy channel model, the probability of a sentence-level analysis is the product of TAG channel model (which generates the reparandum and interregnum) and the parser-based language model (which generates the words that are not in either a reparandum or interregnum). Ideally, we would like to find the strings with highest probability according to this product model. Unfortunately, we don't know an efficient dynamic programming algorithm for searching for these strings, and there may not be one, since the intersection of a TAG (the channel model) and a CFG (the parser-based language model) is in general undecidable.

Instead, we search for the highest scoring analyses using a bigram language model. (Recall that the intersection of a TAG with a FSM is another tree-adjoining language). Specifically, we search for the 25 highest-scoring sentence level analyses using the TAG channel model and a simple bigram language model. The MaxEnt reranker described in section 4 will select one of these as our system's preferred analysis, using the parser-based language model probability in place of the bigram language model probability.

The TAG transducer is trained on the disfluency files in sections 2 and 3 of the version of Switchboard distributed in the Penn Treebank 3 CDROM. This is because the transducer must be trained on data that includes explicitly marked repair strings for each speech repair.

3. THE SYNTACTIC LANGUAGE MODEL

The syntactic language model used here is that described in [2]. It is, in essence, simply a generative probabilistic syntactic parser that when used for parsing searches for the parse that maximizes

$$\arg max_{\pi} p(\pi \mid s) = \arg max_{\pi} p(\pi, s) \qquad (1)$$

where π is a parse tree and s is a sequence of words (a sentence). When used for language modeling we rather attempt to evaluate

$$\sum_{\pi} p(\pi, s) = p(s). \qquad (2)$$

This latter, of course, is simply the required language model.

Also, as noted in the aforementioned paper, the language model version of the parser differs from the parsing version in it incorporates tri-head probabilities (i.e., each head word is conditioned not just on its governor, but on its governor's governor). It is generally recognized that conditioning parsing rules and head selection rules on the governor (i.e., the head of the parent constituent) results in higher

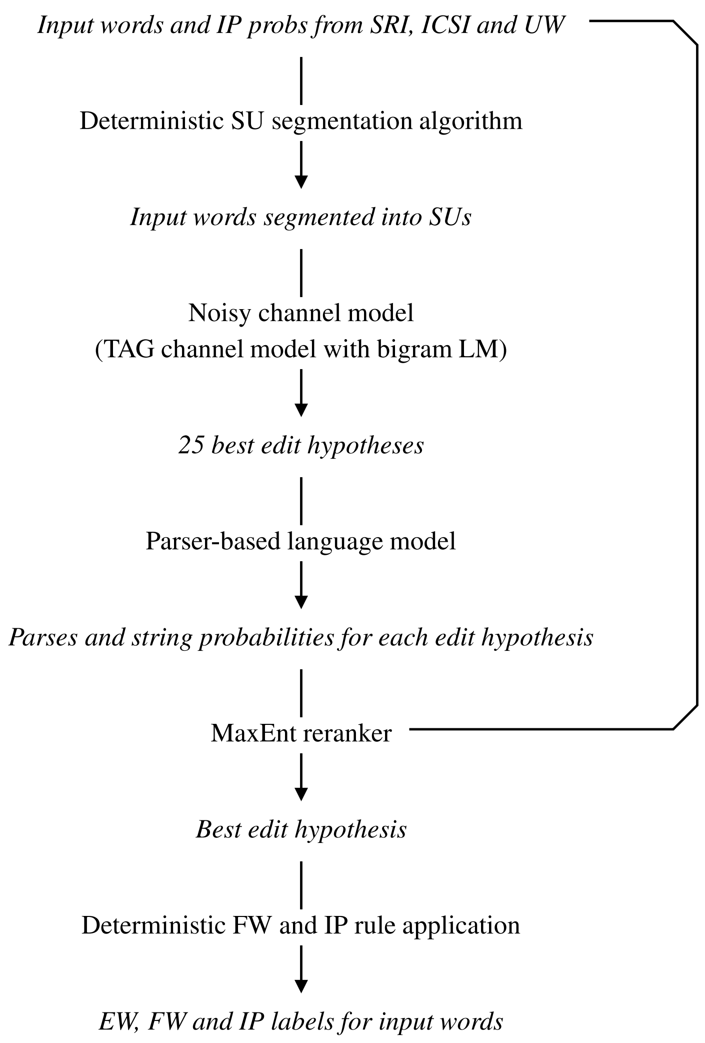
Input words and IP probs from SRI, ICSI and UW
Deterministic SU segmentation algorithm
Input words segmented into SUs
Noisy channel model
(TAG channel model with bigram LM)
25 best edit hypotheses
Parser-based language model
Parses and string probabilities for each edit hypothesis
MaxEnt reranker
Best edit hypothesis
Deterministic FW and IP rule application
EW, FW and IP labels for input words

Fig. 1. The general architecture of our MDE system. Roman script identifies processing steps, while italic script identifies data. The noisy channel model is described in Johnson and Charniak [1], and the parsing language model is described in [7]. The other components are described in this paper.

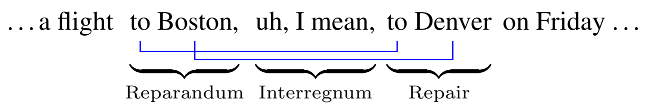
. . . a flight to Boston, uh, I mean, to Denver on Friday . . .
Reparandum
Interregnum
Repair

Fig. 2. The structure of a typical repair, with crossing dependencies between reparandum and repair, following Shriberg [5].

parse accuracy. As one may infer from the (lack of) literature on the topic, conditioning on the governor's governor has no beneficial impact on parsing accuracy (this is what we find). However, this is *not* the case for language modeling. Conditioning head selection on the governor's governor results in a major improvement on perplexity results. This, of course, should not be surprising, since this is the parsing analogue of moving from a bigram language model to a trigram model.

Finally the language model was trained on the tree-banked section of Switchboard included with the Penn Treebank 3 CDROM, along with small sections of less formal text from the Brown Corpus. (Experiments in using Wall Street Journal text resulted in no improvement on switchboard development data, presumably due to the very different makeup of the corpora.)

4. THE MAXENT RERANKER

The MaxEnt reranker is responsible for choosing the best Edit Word hypothesis from the list of the 25 best analyses for each SU generated by the TAG channel model. The primary reason why we used a MaxEnt reranker is that it permits us to tune our model to RT-04F training data (recall that the TAG channel model and the syntactic parser language model are in fact trained on the Switchboard data distributed with the Penn Treebank 3 CDROM). But it also has the added advantage that it permits us to experiment with a wide variety of other features, including features derived from the string and parse tree and features derived from other sources, such as the syntactic context of repairs and prosodic information associated with IPs.

The log probabilities produced by the TAG channel model and the syntactic parser language model are the primary features used by the reranker; if these were the only features, the MaxEnt reranker would essentially implement the standard noisy channel model with adjustable mixing constants for the channel and language models. An advantage of the MaxEnt reranker is that it can incorporate a wide range of different features. In the system described here we added a variety of features based on the local context of the reparandum based on the features we used in an earlier word-by-word Edit Word Detection (EWD) detection system [2]. An informal error analysis of the two EWD algorithms in [2] and [1] suggested that the noisy channel model was better at detecting moderately long speech repairs, but the word-by-word classifier was better at detecting extremely short repairs, so we used many of the features from [2] as features for our MaxEnt model.

We also added two additional types of features. First, since we have syntactic parses of the whole SU, we can identify the syntactic context in which each speech repair occurs, so we experimented with adding features based on the syntactic context here. The features we used were the category labels immediately dominating, preceding and following the repair.

Secondly, we have conducted preliminary experiments using features based on prosodic information, specifically word-by-word IP probabilities provided to us by SRI-ICSI-UW. While we have not conducted a systematic investigation, we found that the most straight-forward way of incorporating these—taking the log probabilities as features—was not successful, so instead we binned the probability values and used each distinct bin as a categorical variable (the feature fires once whenever any word's IP probability falls into the bin). Our suspicion is that the binning helps deal with nonlinearities in the probability estimates of the models involved (i.e., one or both of the log probability distributions estimated by the noisy channel model or by the IP model is nonlinearly related to the true log probability distribution, and this nonlinearity cannot be corrected by a simple scaling constant of the kind that MaxEnt estimates).

5. THE DETERMINISTIC RULES

Our basic system does not return sufficient information to perform the FWD or IPD tasks. We have therefore augmented our base system with a set of hand-written deterministic rules for the FWD task, based on analysis of the Dev1 and Dev2 corpora (see section 6), as well as an earlier analysis of the distribution of disfluencies in speech [8]. IPD predictions were derived from EWD and FWD output.

5.1. Filler Word Detection (FWD)

Filler words are identified and classified into filler types using a small collection of deterministic rules. These rules were constructed as follows. First, we noticed that filler words are approximately distributed in the training data (by token) as follows: 70% Discourse Marker (DM), 30% Filled Pause (FP), $< 1\%$ Explicit Editing Term (EET). Based on this, we decided to ignore EET fillers completely, so our system never predicts a word as EET. We then examined the distribution of FPs and noted that $> 95\%$ of the distribution is composed of the four words "uh", "um", "eh", and "ah". Moreover, we noted that these four words occur almost exclusively as FPs ($> 95\%$ of the time). As a result, we simply label any occurrence of one of these words as a FP, and never predict any other word as a FP. As for DMs, we noted that 2 two-word phrases ("you know", "i mean") and 6 other words ("like", "so", "well", "oh", "actually", "now") comprise 95% of the DMs, so we only label these words as DMs.

This still leaves the question of how we decide whether these phrases are in fact FPs. Our first approach was simple: classify all occurrences of each term according to whichever classification was most frequent for the given term. That is, “you know”, “i mean”, “like”, “so”, and “well” were always labeled DM and no other phrases were labeled DM; this simple approach yields about 30% overall error on the FWD task (on the development corpora). We made a series of small improvements to these basic rules as follows.

First, we label “oh” a DM whenever it was the first word of a sentence and the sentence length was less than or equal to 4. We also label “like” a non-DM if it is preceded by “seem”, “’m”, “feel”, “i”, “n’t”, “something”, “stuff”, “sound”, “things”, “was”, “would”, “you”, “’s” or “’re”, or followed by “that” or “to”. With these two modifications, the deterministic FWD rules have about 22% error.

In addition, we introduced rules sensitive to part-of-speech (POS) labels, as defined by Penn Treebank annotation guidelines [9] and identified by the Charniak parser, into the rules. Specifically, “so” is not labeled DM if it is followed by POS type “IN” or preceded by “AUX” or “RB”, or if the two preceding tokens were both “CC”. Further, “like” is not labeled DM if it is followed by POS types “VBP” or “VB” or preceded by types “VBZ”, “NN”, or “NNS”. With this modification, the deterministic FWD rules have an overall error rate of about 20%.

Finally, we introduced rules that are sensitive to the string’s parse tree. Specifically, “so” is not labeled DM if it is part of an adjectival or adverbial phrase, and “actually” is only labeled DM if it is part of an “UH” phrase or it is uppercase. With these modifications the deterministic FWD rules have an error rate of about 19%. These rules were the ones used to produce the output we submitted for the RT-04F evaluation.

Note that we did not make use of the interregna identified by our EWD system. Interregna account for a small portion of the overall filler phrases, and nearly all of these are correctly identified by the deterministic rules above. Thus even correctly identified interregna would provide negligible improvement to our FWD system.

5.2. Interruption Point Detection (IPD)

Our IP detection is based on the literal definition of Interruption Points: an IP follows each reparandum and precedes each filler phrase [3]. Since our EWD system identifies the end of each reparandum and our FWD system to identifies the start of each filler phrase, we labeled IPs accordingly.

6. INPUT DATA

The following corpora were provided to us as part of the RT-04F MDE evaluation for Conversational Telephone Speech (CTS) [6]:

Dev1 RT-04 MDE DevTest Set #1 V1.2 (LDC2004E16): 72 CTS files (6 hours) drawn from Switchboard (ISIP) and Fisher (http://www.ldc.upenn.edu/Fisher/).

Dev2 RT-04 MDE DevTest Set #2 V1.1 (LDC2004E29): 36 CTS files (3 hours) drawn from Fisher

Evaluation The final evaluation test set was provided directly by NIST: 36 CTS files (3 hours) drawn from Fisher

Training RT-04 MDE Training Data V1.2 (LDC2004E31): 396 CTS files (40 hours) drawn from Switchboard (ISIP).

This data was generated by LDC and provided in RTTM format [6]. Reference lexemes included case information and partial words (RTTM LEXEME sub-type “frag”). While the RTTM format also distinguishes lexeme sub-types further (e.g. proper nouns, acronyms, etc.), we did not make use of this information. Similarly, we did not we utilize any non-lexical information contained in the RTTM format (e.g. event onset and duration times), nor did we use the audio data LDC provided.

The MaxEnt reranker was trained on the Dev1 and Dev2 data sets. We did not use the Training data above because it is not parsed, nor does it have the precise form of disfluency annotations that our base system trains from, although as we explain below, it should be possible to train both the parser and channel model from this data, and we hope to do so in future work.

In addition to these corpora, we used the parsed and disfluency annotated Switchboard data distributed on the Penn Treebank 3 CDROM. As described in the previous sections, the parsed data was used to train the parsing language model, and the Treebank 3 disfluency annotations were used to train the channel model. For testing on automatically recognized STT tokens, partial words and case were stripped out prior to training, but both were left in for testing on manually annotated words.

We were also provided with the following additional data generated by SRI-ICSI-UW [10]: automatically recognized (STT) words, the probability of each lexeme (reference and detected) being immediately followed by an interruption point (IP), and the probability of each lexeme (both reference and detected) ending a sentence-like unit (SU). Recognized lexemes did not include case information or partial words. The IP/SU probabilities were generated using an ensemble of methods. A probabilistic prosody model estimates the conditional probability of an IP/SU event via a decision tree using prosodic features such as duration, fundamental frequency (F0), energy, and pause.

7. RESULTS

In this section we report the performance of our system on the EWD, FWD, and IPD metadata extraction (MDE) tasks

Task	STT	Ref
EWD	76.25%	46.08%
FWD	39.93%	23.69%
IPD	55.88%	28.60%

Table 1. Error rates of our MDE system on the RT-04F Evaluation data.

Features	STT	Ref
all features	75.8%	52.8%
all but IP features	76.4%	54.3%
all but parser LM features	76.7%	55.0%
all but TAG channel features	81.0%	56.5%

Table 2. EWD error rate on the Dev2 data as a function of features used by the MaxEnt reranker.

for conversation telephone speech. For each task, we tested our system on both manually transcribed (Reference) lexemes and automatically recognized Speech-To-Text (STT) lexemes on the Evaluation data. Performance on all tasks was measured using the following simple error metric: the number of mistakes (false positives + false negatives) divided by the actual number of events (true positives).

Official results from the RT-04F blind evaluation are given in Table 1.

To better understand the relative importance of the various components of our EWD system, we performed several contrastive tests in which different subsets of features were omitted from the MaxEnt reranker. The reranker was run with:

1. all features included,
2. all features included except IP features,
3. all features included except Syntactic Language Model probabilities, and
4. all features included except the TAG Channel Model probabilities.

In these tests the reranker was trained on the Dev1 data and evaluated on the Dev2 data, preserving the Evaluation data as a final test corpus for later work. The results of these tests are given in Table 2.[1]

Since our EWD system depends heavily on input detected SU boundaries, we were also interested in examining the effect of SU detection error on our EWD error rate. To measure this, we evaluated our EWD system on the Dev2 data using reference SU boundaries. Using reference SU boundaries our system has a 44.3% EWD error rate, which is a 16% error rate reduction compared to the 52.8% EWD error rate our system obtains using automatically detected SU boundaries.

[1]Note that the MaxEnt reranker's features in our system were chosen to complement the parser based language model and the TAG channel model, and were not changed during these tests. It may be able to improve performance of a system without these components by using other kinds of MaxEnt features.

8. CONCLUSION AND FUTURE WORK

The noisy channel model of speech repairs we proposed previously [1] was extended with a maximum entropy reranker and applied to the task of Edit Word Detection (EWD). We also augmented this system with a set of manually constructed deterministic rules for Filler Word Detection (FWD), and the combination of our EWD and FWD sub-systems enabled us to also predict self-interruption points (IPs). We evaluated our system for these three tasks on two types of input: manually annotated words and automatically recognized speech-to-text tokens. In all six cases, our system improves the state-of-the-art as measured on the recent RT-04F evaluation. Perhaps the most encouraging result of this evaluation was that the resulting system was highly competitive on speech recognizer output as well as human-produced transcripts. That it would work well on speech recognizer output was not a given, as the system uses a parser-based syntactic language model which might have been thrown off by incorrectly recognized words anywhere in the sentence.

We envision significant future work that we hope will result in improved system performance. Roughly speaking this work falls into two groups: first, work on better fitting the system to the RT-04F tasks that we were not able to accomplish (owing to our late decision to enter the RT-04F competition) and second, true future research. We discuss these in turn.

8.1. Future Task Adaptation

In many cases the data used for training our models was mismatched to the task. To take the most blatant example, both the channel model and language model were trained on Penn Treebank 3 version of Switchboard, as opposed to the more recent Training corpus. Since the Treebank 3 Switchboard transcription conventions, sentence segmentations and repair annotations differ from those used in the RT-04F corpora, training from RT-04F corpora should decrease the error rate of our system.

We would like to incorporate the Filler Word Detection task into the noisy channel model. Based on previous research [8] we believe that filler words make parsing slightly more difficult (at least in terms of parsing accuracy). Thus having the channel model remove filler words before the syntactic language model is used to parse the remaining words would seem to make sense.

In the system used here, our MaxEnt model was trained on a comparatively small amount of data (the Dev1 and Dev2 data sets). This effectively makes it impossible to use infrequently occurring features, such as lexicalized features, since there simply isn't enough training data to accurately estimate the weights in the MaxEnt model for such features. The much larger Training corpus is an obvious thing to use.

Of course, the MaxEnt training data must be processed by the TAG channel model and each of the 25-best analyses for each string must be parsed by the parser-based language model. (Moreover, if this training data is to be used to train other components of the system, this must be done in an n-fold training setting [4]). This may require parsing several millions of sentences, which is only practical using a compute cluster.

8.2. Future Research

Given the encouraging results of applying our model to the RT-04F tasks we competed in, we look forward to extending it to other, related, problems.

Perhaps the most immediately obvious is the SU boundary detection problem, the only one of RT-04F MDE tasks in which we did not participate. We expect that the string-nonlocal dependencies that our parser-based language model is sensitive to should be of use in determining where sentences begin and end. As noted above, we found that using transcript SU boundaries reduces our system's EWD error rate, so improved SU detection should synergistically reduce EWD error rate as well.

We would also like to work directly off the word lattices produced by speech recognition systems. Keith Hall has shown that parser-based language models can work directly with speech recognizer lattices [11], and it may be possible to develop a version of our EWD system that does this too.

Finally we would like to explore the use of partially labeled, and unlabeled, training data for these tasks. In particular, there is a common misperception that syntactic language models require hand parsed data for training, thus limiting how much data can be used in their creation. This is not the case. While it is true that unparsed data does not seem to help parsing accuracy, we have noticed that additional unparsed training data does improve both the perplexity and word-error rates of our parser-based language model. We intend to investigate this and other uses of unlabeled data to improve our results.

9. REFERENCES

[1] Mark Johnson and Eugene Charniak, "A TAG-based noisy channel model of speech repairs," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, 2004, pp. 33–39.

[2] Eugene Charniak and Mark Johnson, "Edit detection and parsing for transcribed speech," in *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*. 2001, pp. 118–126, The Association for Computational Linguistics.

[3] "Simple metadata annotation specification version 6.2," Tech. Rep., Linguistic Data Consortium, 2004, Available at http://www.ldc.upenn.edu/Projects/MDE.

[4] Michael Collins, "Discriminative reranking for natural language parsing," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, Stanford, California, 2000, pp. 175–182.

[5] Elizabeth Shriberg, *Preliminaries to a Theory of Speech Disfluencies*, Ph.D. thesis, University of California, Berkeley, 1994.

[6] National Institute of Standards and Technology, "2004 rich transcription (RT-04F) evaluation plan, version 14," Tech. Rep., Available at http://www.nist.gov/speech/tests/rt/rt2004/fall, Visited 10/30/04.

[7] Eugene Charniak, "Immediate-head parsing for language models," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. 2001, The Association for Computational Linguistics.

[8] Donald Engel, Eugene Charniak, and Mark Johnson, "Parsing and disfluency placement," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 2002, pp. 49–54.

[9] Beatrice Santorini, "Part-of-speech tagging guidlines for the penn treebank project (3rd revision)," Tech. Rep., University of Pennsylvania, 1990, Available at http://www.cis.upenn.edu/ treebank/home.html.

[10] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, B. Peskin, and M. Harper, "The ICSI-SRI-UW metadata extraction system," in *Proceedings of the International Conference on Spoken Language Processing*, 2004.

[11] Keith Hall and Mark Johnson, "Attention shifting for parsing speech," in *The Proceedings of the 42th Annual Meeting of the ACL*, 2004.