Recognizing Disfluencies in Conversational Speech

Matthew Lease, *Student Member, IEEE*, Mark Johnson, and Eugene Charniak

***Abstract*—We present a system for modeling disfluency in conversational speech: repairs, fillers, and self-interruption points (IPs). For each sentence, candidate repair analyses are generated by a stochastic tree adjoining grammar (TAG) noisy-channel model. A probabilistic syntactic language model scores the fluency of each analysis, and a maximum-entropy model selects the most likely analysis given the language model score and other features. Fillers are detected independently via a small set of deterministic rules, and IPs are detected by combining the output of repair and filler detection modules. In the recent Rich Transcription Fall 2004 (RT-04F) blind evaluation, systems competed to detect these three forms of disfluency under two input conditions: a best-case scenario of manually transcribed words and a fully automatic case of automatic speech recognition (ASR) output. For all three tasks and on both types of input, our system was the top performer in the evaluation.**

***Index Terms*—Disfluency modeling, natural language processing, rich transcription, speech processing.**

I. Introduction

In spontaneous speech, speakers form their utterances on the fly and subject to various cognitive constraints and pragmatic factors. As a result, such speech is often *disfluent*, containing partial words (or word fragments), fillers[1] (e.g., `uh`, `um`, `you know`, `like`, etc.), and repairs, such as that shown in Fig. 1. While the presence of disfluencies rarely poses any comprehension problem to an engaged listener, they have been shown to negatively impact both the readability of transcribed speech [2] and the accuracy of automated analysis performed on it [3]. Such findings demonstrate the importance of moving beyond bare stream-of-words automatic speech recognition (ASR) to instead generate *enriched transcriptions*, in which disfluencies and sentence[2] boundaries are recognized in addition to words. This line of research has been notably pursued in recent years via the Defense Advanced Research Projects Agency (DARPA) EARS program.

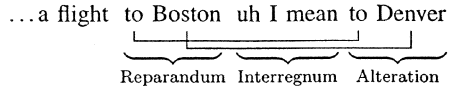


Fig. 1. Structure of a typical repair, with crossing dependencies between reparandum and alteration.

In previous work, we presented two systems for detecting speech repairs [3], [4], showing that a noisy-channel model based on a tree adjoining grammar (TAG) outperformed an earlier word-based classifier. In this paper, we add support for filler and self-interruption point (IP) detection and further improve repair detection via maximum-entropy modeling.

The overall architecture of our system is sketched in Fig. 2. The TAG model (Section II) takes tokenized[3] input sentences and proposes candidate repair analyses, where each analysis identifies zero or more hypothesized repair locations for a given sentence. For each analysis, the fluency of the remaining, nonrepair words in the sentence is scored using a syntactic language model (Section III). Given these analyses and their associated scores, a maximum-entropy classifier (Section IV) is then used to rerank the hypotheses and select the most probable one. Use of maximum-entropy modeling permits us to use a wide range of additional information to identify the best sentence-level analysis, something that would be harder to realize in the first-stage generative model.

A limitation of the system described above is that the only form of disfluency modeled is speech repairs. While the TAG also identifies fillers involved in speech repairs, most fillers actually occur outside of repair contexts and so are not identified by that model. To address this, we augmented our TAG-based model with a small set of hand-crafted, deterministic rules for filler detection (Section V). While our eventual goal is to incorporate filler detection directly into the noisy-channel model, these simple rules have proven to be surprisingly effective. Support for IP detection was added via deterministic combination of repair and filler predictions (Section VII).

Section VI describes data used in training and evaluating our system, and Section VII presents system performance in the Rich Transcription Fall 2004 (RT-04F) blind evaluation [5]. As part of this evaluation, system performance was measured on two types of input: a best-case condition of manually transcribed words and a fully automatic case of ASR output. For all three tasks and on both types of input, our system was the top performer in the evaluation. The results on ASR output are particularly encouraging in showing that syntactic language modeling features continue to be useful even in the presence of significant word recognition errors (Table II).

Manuscript received September 30, 2005; revised April 26, 2006. This work was supported by the National Science Foundation under Grants LIS 9720368 and IIS0095940. A version of this paper was presented at the Rich Transcription Fall 2004 Workshop (RT-04F). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. John Makhoul.

M. Lease and E. Charniak are with the Department of Computer Science, Brown Laboratory for Linguistic Information Processing (BLLIP), Brown University, Providence, RI 02912 USA (e-mail: mlease@cs.brown.edu, ec@cs.brown.edu).

M. Johnson is with the Department of Cognitive and Linguistic Sciences, Brown Laboratory for Linguistic Information Processing (BLLIP), Brown University, Providence, RI 02912 USA (e-mail: mj@cs.brown.edu).

Digital Object Identifier 10.1109/TASL.2006.878269

[1]While the term *filler* has traditionally been synonymous with *filled pause*, we adopt recent Linguistic Data Consortium (LDC) SimpleMDE [1] conventions in using the term to describe a broad set of vocalized space-fillers which includes filled pauses.

[2]We use *sentence* and *sentence-like unit* (SU) interchangeably in this text. While SU is more precise [1], the notion of sentences is widely understood.

[3]http://www.cis.upenn.edu/~treebank/tokenizer.sed

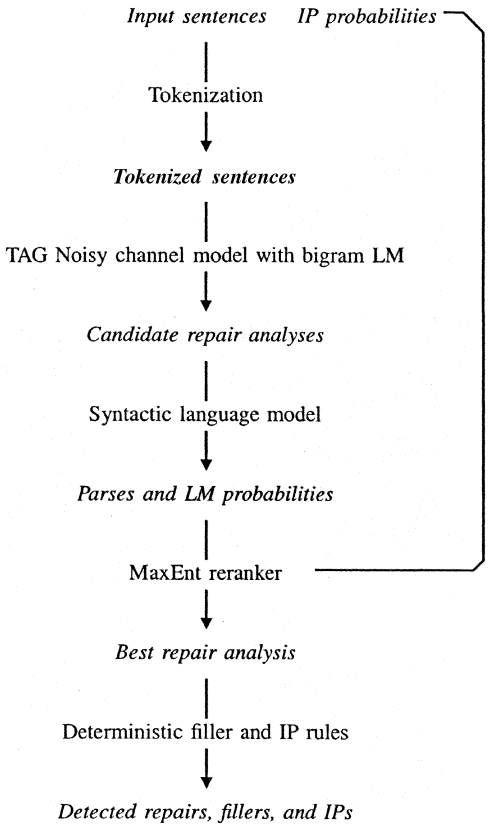


Fig. 2. Overall system architecture. Roman script identifies processing steps, while italic script identifies data. Input IP probabilities serve as additional features for reranking (Section IV).

II. TAG Channel Model

The TAG model is responsible for proposing candidate repair analyses for each sentence. We follow previous work on speech repairs [6], [7] in dividing a repair into three parts: the *reparandum* (the material repaired), the *interregnum* (or editing term) that typically consists of zero or more fillers, and the *alteration*. Fig. 1 shows these three parts for a typical repair. As this figure shows, the alteration can often be understood as a *rough copy* of the reparandum, using the same or very similar words in roughly the same order [7]. In other words, a speech repair seems to involve cross-serial dependencies between the reparandum and the alteration, as shown in Fig. 1. Languages with an unbounded number of cross-serial dependencies cannot be described by a context-free or finite-state grammars, and crossed dependencies like these have been used to argue natural languages are not context-free languages [8]. Fortunately, mildly context-sensitive grammar formalisms such as TAGs and combinatory categorial grammars can model cross-serial dependencies, and this is why we adopt a TAG-based model here.

Fig. 3 shows our TAG model's dependency structure for the repair of Fig. 1. Interestingly, if we trace the temporal word string through this dependency structure, aligning words next to the words they are dependent on, we obtain a "helical" type of structure familiar from genomics, and in fact TAGs are being used to model genomes for very similar reasons [9].

To effectively model both the crossed-dependencies of repairs and the more usual linear or tree-structured dependencies

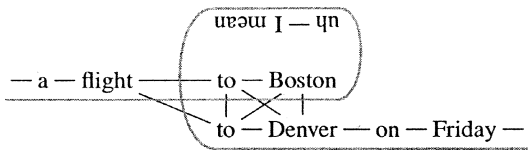


Fig. 3. The "helical" dependency structure induced by the generative model of speech repairs for the repair depicted in Fig. 1.

of nonrepaired speech, we adopt the noisy channel paradigm [7], [10]. We begin by imagining that speakers intend to say source sentences X (with no repairs), but may mistakenly insert one or more repairs, producing observed sentences Y. Our goal is for each observed sentence to recover the most likely source sentence $\widehat{x}$. Applying Bayes Rule, we can formulate this problem in canonical noisy channel form

$$\widehat{x} = \operatorname*{argmax}_X P(X|Y) = \operatorname*{argmax}_X P(Y|X)P(X).$$

The *channel model* $P(Y|X)$ defines a stochastic mapping of source sentences into observed sentences via the optional insertion of one or more repairs, and the *language model* $P(X)$ defines a probability distribution over source sentences. This is the same general setup that has also been effectively applied to speech recognition [11] and machine translation [12]. In our application, the channel model is realized as a TAG, and we train our syntactic language model (Section III) on sentences with the speech repairs removed.

The channel model nondeterministically predicts repairs at every position in the input string, conditioned on the preceding word. Because of this conditioning, 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 it and the word in the alteration string corresponding to it. The TAG transducer is effectively a simple first-order Markov model (albeit one that captures string nonlocal dependencies).

A. Formal Description

The TAG channel model defines a stochastic mapping of source sentences X into observed sentences Y. There are several ways to define transducers using TAGs such as [13], but the following simple method inspired by finite-state transducers suffices for the application here. The TAG defines a language whose vocabulary is the set of pairs $(\Sigma \cup \{\emptyset\}) \times (\Sigma \cup \{\emptyset\})$, where Σ is the vocabulary of the observed sentences Y. A string Z in this language can be interpreted as a pair of strings (Y, X), where Y is the concatenation of the projection of the first components of Z and X is the concatenation of the projection of the second components. For example, the string $Z =$ *a: a flight:flight to:∅ Boston: ∅ uh: ∅ I: ∅ mean: ∅ to:to Denver:Denver on:on Friday:Friday* has the observed string $Y =$ *a flight to Boston uh I mean to Denver on Friday* and the source string $X =$ *a flight to Denver on Friday*. Fig. 4 shows the TAG rules used to generate this example.

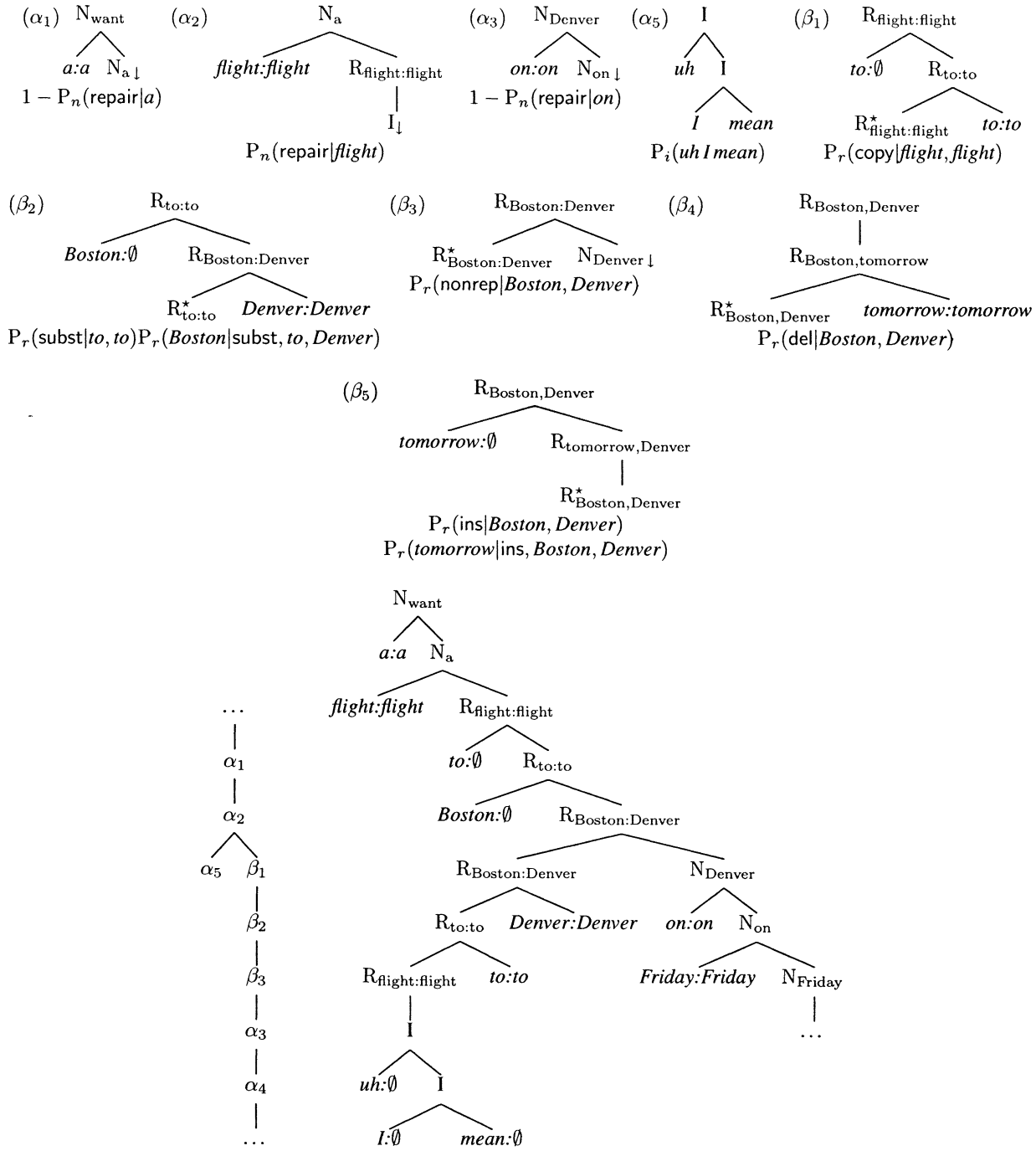


Fig. 4. TAG rules used to generate the example shown in Fig. 1, their respective weights, and the corresponding derivation and derived trees. The nonterminals in this grammar are of the form N_{w_x}, $R_{w_y:w_x}$, and I, where w_x is a word appearing in the source string, and w_y is a word appearing in the observed string. Informally, the N_{w_x} nonterminals indicate that the preceding word w_x was analyzed as not being part of a repair, while the $R_{w_y:w_x}$ that the preceding words w_y and w_x were part of a repair with w_x in the alteration corresponding to w_y in the reparandum. The nonterminal I generates words in the interregnum of a repair. $R^\star$ indicates the foot node in an auxiliary (repair) tree. Encoding the preceding words in the TAG's nonterminals permits the channel model to be sensitive to lexical properties of the preceding words. The start symbol is $N_\$$, where "$" is a distinguished symbol used to indicate the beginning and end of sentences. Details of the associated conditional probability distributions above can be found in [4].

B. Model Estimation

To estimate weights on the TAG productions described in Section II-A, we need to know the TAG derivation of each sentence in the training data. Uniquely determining this requires knowing both the locations of each reparandum, interregnum, and alteration regions of each repair (which are annotated—see Section VI) and the crossing dependencies between the reparandum and alteration words (as indicated in Fig. 1), which are not annotated. We obtain the latter by aligning the reparandum and repair strings of each repair using a minimum-edit distance string aligner like [14] with the following alignment costs:

0 identical words;
2 words with the same POS tag;
4 an insertion or a deletion;
5 words with POS tags that begin with the same letter;
7 an arbitrary substitution.

These costs were chosen so that a substitution will be selected over an insertion followed by a deletion, and the lower cost for substitutions involving part-of-speech (POS) [15] tags beginning with the same letter is a rough and easy way of establishing a preference for aligning words whose POS tags come from the same broad class, e.g., it results in aligning singular and plural nouns, present and past participles, etc. While we did not quantitatively evaluate the quality of the alignments (since they are not in themselves the object of this exercise), manual inspection convinced us they were sufficiently accurate for our purposes.

From our training data, we estimate a number of conditional probability distributions (see [4] for full details). These estimated distributions are the linear interpolation of the corresponding empirical distributions from the main subcorpus using various subsets of conditioning variables (e.g., bigram models are mixed with unigram models, etc.) using Chen's bucketing scheme [16]. As is commonly done in language modeling, the interpolation coefficients are determined by maximizing the likelihood of the heldout data counts using expectation maximization (EM). Special care was taken to ensure that all distributions over words ranged over (and assigned nonzero probability to) every word that occurred in the training corpora; this turns out to be important as the size of the training data for the different distributions varies greatly.

C. Computation

While we assume that the source sentence X is a substring of the observed string Y, the number of possible Xs grows exponentially with the length of Y, so exhaustive search is probably infeasible. However, polynomial-time dynamic programming parsing algorithms available for TAGs can be used to search for repairs. To use these algorithms, it is necessary that the intersection (in language terms) of the TAG channel model and the language model itself be describable by a TAG. One way to guarantee this is to use a finite-state language model, so we use a bigram language model with the TAG to generate candidate analyses for each sentence. The syntactic language model is then run on the nonrepaired portions of these to refine the bigram language model scores.

We use the standard bottom-up dynamic programming TAG parsing algorithm to search for candidate parses. Because our TAG only involves adjunction of single words rather than arbitrary phrases, the general n^6 TAG parsing algorithm runs in n^5 time, where n is the length of the string. There is an additional optimization we use to reduce running time. Even though our sentences are often long, it is extremely unlikely that any repair will be longer than, say, 12 words. So to increase processing speed, we only compute analyses for strings of length 12 or less. For every such substring that can be analyzed as a repair, we calculate the *repair odds*, i.e., the probability of generating this substring as a repair divided by the probability of generating this substring via the nonrepair rules, or equivalently, the odds that this substring constitutes a repair. The substrings with high odds are likely to be repairs.

This more local approach has a number of advantages over computing a global analysis. First, as just noted it is much more efficient to compute these partial analyses rather than global analyses of the entire sentence. Second, there are rare cases in which the same substring functions as both alteration and reparandum (i.e., the alteration is also repaired). A single global analysis would not be able to capture this (since the TAG channel model does not permit the same substring to be both a reparandum and an alteration), but we combine these overlapping repair substring analyses in a post-processing operation to yield an analysis of the whole sentence. (We do insist that the reparandum and interregnum of a repair do not overlap with those of any other repairs in the same analysis).

III. Syntactic Language Model

Given a set of candidate repair analyses for each utterance, the syntactic language model is used to score the fluency of the remaining nonrepair words under each analysis. To this end, we apply the generative syntactic probabilistic model described completely in [17]. When used for parsing, the model seeks to maximize

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

where π is a parse tree and s is a sequence of words (an utterance). For language modeling, rather than simply finding the most probable parse, instead we sum the probability mass over the set of most likely parse trees

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

to yield $p(s)$, the language model score for the utterance.

IV. Maximum-Entropy Reranker

The maximum-entropy reranker is responsible for choosing the best repair analysis for each sentence from a set of hypotheses generated by the TAG channel model. Reranking has the 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. Reranking also allowed us a simple means of addressing the mismatch between the training and evaluation data used in our experiments (see Section VI).

The general setup employed here is the same as we used in parse reranking [18]. The reranker selects from a set of analyses $T = \{t_1, \ldots, t_k\}$ the analysis $t^\star \in T$ with the maximum score. A feature-extractor converts each analysis into a vector of real-valued features $f(t) = (f_1(t), \ldots, f_m(t))$ (e.g., the value $f_j(t)$ of the feature f_j could be the log probability of the analysis t under the syntactic language model). Each feature f_j is associated with a real-valued weight λ_j, and $\lambda \cdot f(t)$ (the dot product of the feature vector and the weight vector λ) is a single scalar score for each analysis. The reranker employs a maximum-entropy estimator that selects the λ that minimizes the log loss of the highest scoring analysis $t^\star$ conditioned on T (together with a Gaussian regularizer to prevent over-training). Informally, λ is chosen to make the correct analyses as likely as possible under the (conditional) distribution defined by f and λ.

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-based repair detector [3]. An informal error analysis of the two detection algorithms in [3] and [4] suggested that the noisy channel model was better at detecting moderately long speech repairs, but the word-based classifier was better at detecting extremely short repairs, so we used many of the features from [3] as features for our MaxEnt model.

We also added two additional types of features: syntactic and prosodic. The syntactic features were the category labels immediately dominating, preceding, and following the repair. The prosodic features consisted of per-word IP probabilities (see Section VI). We found that the most straight forward way of incorporating the prosodic features—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). We suspect 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 and/or IP models 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).

V. Filler Word Detection

SimpleMDE [1] annotates three filler types to be recovered.

- **Filled Pauses (FPs)**: *hesitation sounds that speakers employ to indicate uncertainty or to maintain control of a conversation while thinking of what to say next*. Examples include `ah`, `eh`, `er`, `uh`, `um`, etc.
- **Discourse Markers (DM's)**[4]: *words or phrases that function primarily as structuring units of spoken language*. Examples include `you know`, `i mean`, `like`, `so`, `well`, `oh`, etc.
- **Explicit Editing Terms (EETs)**: *overt statements from the speaker recognizing the existence of an edit disfluency*. Examples include `i mean`, `or rather`, etc.

As shown in Fig. 2 and described in Section VI, input to our filler detection component consists of tokenized words segmented by detected sentence boundaries. As described below, we also exploit detected POS and syntactic information provided by the syntactic language model (Section III), which outputs the most likely parse tree as well as the language model score for each repair analysis candidate generated by the TAG (Section II). Because EETs account for less that 1% of all filler words, as measured in the development section of our data (Section VI), we do not model them in our system.

[4]While SimpleMDE's definition of *discourse marker* is similar to that used in discourse parsing literature [19], [20], the examples (and data) suggest that what is being annotated is actually quite different. Rather than indicate discourse structure, these items appear to function similarly to filled pauses.

FPs comprise about 30% of all filler words, and 95% of these consist of `ah`, `eh`, `uh`, and `um`. These words are always labeled as FPs by our system. We do not model the remaining 5% of FPs, which consist of unusual orthography and words which more often function as nonfillers, other filler types, or backchannels (i.e., words like `uh-huh` used to indicate the listener is still engaged in the conversation).

DMs account for the remaining 70% of filler words. Like FPs, DMs also have a rather peaked distribution: `you know`, `i mean`, `like`, `so`, `well`, `oh`, `actually`, `and now` comprise 95% of all DMs. However, unlike FPs, these terms also occur frequently as nonfillers:

- `do you know` what `i mean`?
- he is `so` busy, `like` many who `actually` work.

As a result, classifying the terms above by their most frequent labeling (DM or nonfiller) and detecting FPs as described earlier only achieves a filler word detection (FWD) error of about 30%, where error is defined as the number of misclassifications divided by the number of true filler words. To improve upon this, a few simple lexical, POS, and syntactic rules were adopted, as listed below.

Lexical rules reduced overall error to about 22%, POS rules to about 20%, and syntactic rules to about 19%.

Lexical rules

- **like**: label as nonfiller if (a) preceded by `'m`, `'re`, `'s`, `feel`, `i`, `n't`, `seem`, `something`, `sound`, `stuff`, `things`, `was`, `would`, or `you` or (b) followed by `that` or `to`.
- **oh**: label as DM whenever it is not the first word of a sentence or the sentence is longer than four words.

POS rules

- **like**: label as nonfiller if (a) followed by `VB` or `VBP` or (b) preceded `NN`, `NNS`, or `VBZ`.
- **so**: label as nonfiller if followed by (a) `IN`, (b) preceded by `AUX` or `RB`, or (c) if the two preceding tokens were both `CC`.

Syntactic rules

- **actually**: label as DM *only* if it is either part of an interjection (UH) phrase or if it begins the utterance
- **so**: label as nonfiller if part of an adjectival (ADJP) or adverbial (ADVP) phrase.

As a final note, recall our earlier comment that the TAG model identifies fillers involved in speech repairs, but that most fillers actually occur outside of repair contexts. Because the deterministic rules above worked well in both repair and nonrepair contexts, we found that even oracle detection of fillers in repair contexts could only negligibly improve overall performance. Therefore, we discarded filler predictions made by the TAG and predicted fillers entirely on the basis of the rules described above.

VI. Input Data

Our system was trained and evaluated on conversational telephone speech drawn from Switchboard [21] and Fisher:[5]

- TB3 Penn Treebank-3 (LDC99T42): contains 1126 disfluency (Meteer) annotated [22] Switchboard conversations, 650 of which were treebanked [23], [24].

[5]http://www.ldc.upenn.edu/Fisher

- Dev1 RT-04 MDE DevTest Set #1 V1.2 (LDC2004E16): 72 SimpleMDE [1] Switchboard and Fisher conversations (6 h).
- Dev2 RT-04 MDE DevTest Set #2 V1.1 (LDC2004E29): 36 SimpleMDE Fisher conversations (3 h).
- Eval RT-04 MDE Eval Data V1.1 (LDC2004E50): 36 SimpleMDE Fisher conversations (3 h).

We did not make use of the RT-04 MDE Training Data (LDC2004E31) due to logistics of our participation in the evaluation. There are important differences worth noting between the annotation schemes used in these various corpora, particularly in terms of disfluencies and sentence boundaries. Our goal was to automatically recover disfluency annotations consistent with SimpleMDE guidelines [1]. As such, one challenge of our work was determining how to make best use of training data annotated according to other schemes, and this is discussed further next.

The TAG transducer (Section II) was trained on the Sections II and III of the Meteer-annotated Switchboard corpus (932 conversations), with Section IV (194 conversations) reserved for future use. Meteer markup [22] was used for training instead of SimpleMDE-annotated data because TAG training requires the location of the *alteration* portion of speech repairs not annotated under SimpleMDE. Section VIII describes future work to facilitate use of SimpleMDE data for TAG training.

The syntactic language model (Section III) was trained on Sections II and III of the Switchboard treebank (496 conversations), with Section IV (154 conversations) reserved for future use. Small sections of less formal text from TB3's Brown corpus were also found to be useful when included in training (though use of TB3's *Wall Street Journal* text was found to have no statistically significant effect). Another case of annotation standard mismatch occurred here in conflicting notions of sentence boundaries between TB3 syntax annotation and SimpleMDE. This may be addressed in future work through use of the newly released treebank of the RT-04 MDE development and evaluation data (LDC2005E15), and through automatic parsing of the RT-04 MDE Training data.

Once trained, the TAG and language model were run on `Dev1`, `Dev2`, and `Eval` to generate scored analyses for reranking (Section IV). During development, the reranker was trained on `Dev1` and tested on `Dev2`. For the final system run, the reranker was trained on both `Dev1` and `Dev2` and tested on `Eval`. As mentioned in Section IV, one motivation for using MaxEnt is that it helps us address the annotation mismatch between training and evaluation data. While we do not report results here, we did find that training the reranker on `Dev1` alone significantly outperformed cross-validated training on TB3 Switchboard, despite the much larger size of the latter.

For corpus preparation, partial words and capitalization were stripped out of training data for ASR evaluation but preserved when testing on manually transcribed words (per RT-04F guidelines [5]). We did not use word-type information (e.g., proper noun, acronym, etc.) or temporal information (e.g., event onset and duration times) that was available in the SimpleMDE rttm data. We also made no direct use of the audio data, though it was used indirectly through our collaboration with SRI-ICSI-UW for the RT-04F evaluation.

TABLE I
ERROR RATES OF OUR MDE SYSTEM ON THE RT-04F EVALUATION TASKS GIVEN REFERENCE AND ASR INPUT WORDS

Input	Task	Del	Ins	Sub	Total
Ref	EWD	36.13	9.95	-	46.08
	FWD	15.49	8.04	0.16	23.69
	IPD	19.21	9.40	-	28.60
ASR	EWD	67.93	8.32	-	76.25
	FWD	25.43	13.78	0.71	39.93
	IPD	40.84	15.04	-	55.88

As part of this collaboration, we were provided with ASR output, detected s SU boundaries for both ASR and manual-transcription conditions, and per-word IP probabilities (the probability of each word being followed by an IP event). The IP/SU probabilities were generated by an ensemble of decision tree, HMM, and maximum entropy models that leveraged lexical, part-of-speech, and prosodic features [25]. For our part, whereas the ASR words and SU boundaries represented fundamental inputs to our system (i.e., we take as input words segmented into SUs), the IP probabilities simply provided an additional feature for reranking (Section VI). Their benefit is shown in Table II.

VII. RESULTS

In this section, we report the performance of our system on the following metadata detection tasks measured in the Rich Transcription 2004 Fall (RT-04F) blind evaluation.

- **Edit Word Detection** (EWD): Label each token as to whether or not it occurs in a speech repair reparandum.
- **Filler Word Detection** (FWD): Label each token as to whether or not it occurs as part of a filler. Filler-type must also identified: FP, DM, or explicit editing term (EET). These types are described with examples in Section V.
- **Interruption Point Detection** (IPD): Label each inter-word gap as to whether or not speech becomes disfluent at that point. An IP occurs whenever the previous word ends a reparandum or the next word begins a filler.

In terms of IP detection, since our EWD system identifies the end of each reparandum region and our FWD system identifies the start of each filler phrase, we were able to predict IPs by simply taking the union of our EWD and FWD predictions.

For all tasks, the goal was to detect disfluency consistent with SimpleMDE annotations [1], where error was measured as the number of misclassifications divided by the number of true events (true positives). For each task, system performance was measured on both the best-case of manually transcribed (reference) words and the fully automatic case of ASR output. For all three tasks and on both types of input, our system was the top performer in the evaluation. Results of our system performance are given in Table I.

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 used:

1) all features included (full system);
2) all features except the IP features;
3) all features except the Language Model probabilities;

TABLE II
EWD ERROR RATE ON THE DEV2 DATASET AS A FUNCTION OF FEATURES USED BY THE MAXIMUM-ENTROPY RERANKER (ALTERNATELY EXCLUDING IP, SYNTACTIC LM, AND TAG CHANNEL FEATURES)

Input	Feats	Del	Ins	Total
Ref	all	44.22	8.57	52.79
	all but IP feats	45.53	8.76	54.28
	all but LM feats	46.09	8.94	55.03
	all but TAG feats	46.90	9.57	56.47
ASR	all	66.34	9.41	75.76
	all but IP feats	66.13	10.22	76.35
	all but LM feats	68.03	8.63	76.66
	all but TAG feats	69.96	11.03	80.99

4) all features except the TAG Channel Model probabilities.

In these tests, the reranker was trained on the Dev1 data and evaluated on the Dev2 data (we reserved further use of the `Eval` test set for final runs in future work). The results of these tests are given in Table II.[6]

Since our noisy-channel repair detection module depends heavily on input sentence boundaries, we were also interested in examining the effect of boundary detection error on its performance. Using reference sentence boundaries, our system achieves a 44.3% EWD error rate, which is a 16% error rate reduction over the 52.8% EWD error rate seen using automatically detected sentence boundaries. This indicates our repair detection module stands to gain significantly from future advances in sentence boundary detection.

VIII. CONCLUSION AND FUTURE WORK

We extended a TAG-based model of speech repairs [4] with a maximum-entropy reranker. We also augmented this system with a set of manually constructed deterministic rules for detecting fillers and showed that repair and filler predictions could be combined to predict self-interruption points (IPs) as well. System performance on these three tasks was measured on two types of input: a best-case scenario of manually transcribed words and a fully automatic case of ASR output. In all six cases, our system improved the state-of-the-art, as measured in the recent RT-04F evaluation. As noted earlier, we found the results on ASR output to be particularly encouraging in showing that syntactic language modeling features continue to be useful even in the presence of significant word recognition errors (Table II). Our existing system also stands to benefit from additional data not used in this evaluation: the RT-04 MDE Training Data (Section VI) and the new RT-04 MDE development and evaluation data treebank (LDC2005E15).

In regard to future work, we believe filler detection can be improved by incorporating filler generation directly into the TAG-based noisy-channel model. In addition to improving Rich Transcription [5], there is strong evidence that improved filler detection can improve parsing [26], and we expect other automated processing of transcribed speech (e.g., machine translation) to benefit similarly. On a related note, since little cross-linguistic work has been done to date in evaluating the relationship between disfluency detection and parse accuracy, we are very interested in evaluating the performance of our system on other languages. Such work is needed to help us understand the extent to which existing techniques and findings our applicable beyond English.

We would also like to explore the use of partially labeled and unlabeled training data. In terms of language modeling, there is a common misperception that syntactic language models can only learn from hand annotated examples, thus restricting their ability to scale up to the amount of data commonly used with today's best n-gram based models. We have already shown, however, that additional unannotated data *can* be used to improve both the perplexity and word-error rates of our syntactic language model [11]. Moreover, a new method of semisupervised parsing has demonstrated significantly improved parsing accuracy over the previous state-of-the-art, and this in turn will directly improve our ability to leverage unlabeled data in syntactic language modeling [27]. In terms of the TAG channel model, recall that its training requires knowing the location of the alteration region of speech repairs, which is not annotated under SimpleMDE. We believe an EM-based approach can be applied to automatically locate these regions by constraining the TAG to generate the alteration given the annotated reparandum and interregnum regions. In addition to improving the accuracy of our repair modeling, the ability to automatically locate alterations will be generally useful for automated processing of speech.

[6]Note that the 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 features.

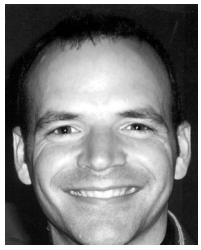
ACKNOWLEDGMENT

The authors would like to especially acknowledge the contributions of M. Ostendorf, Y. Liu, D. Hillard, and J. G. Kahn.

REFERENCES

[1] "Simple Metadata Annotation Specification Version 6.2," Linguistic Data Consortium, Philadelphia, PA, 2004.
[2] D. Jones, F. Wolf, E. Gibson, E. Williams, E. Fedorenko, D. Reynolds, and M. Zissman, "Measuring the readability of automatic speech-to-text transcripts," in *Proc. Eurospeech*, 2003, pp. 1585–1588.
[3] E. Charniak and M. Johnson, "Edit detection and parsing for transcribed speech," in *Proc. 2nd Meeting North Amer. Chap. Assoc. Comput. Ling.*, 2001, pp. 118–126.
[4] M. Johnson and E. Charniak, "A tag-based noisy channel model of speech repairs," in *Proc. 42nd Annu. Meeting Assoc. Comput. Ling.*, 2004, pp. 33–39.
[5] (2004) Rich Transcription (RT-04F) Evaluation Plan, Version 14, Fall 2004. Nat. Inst. Standards Technol., Gaithersburg, MD. [Online]. Available: http://www.nist.gov/speech/tests/rt/rt2004/fall
[6] E. Shriberg, "Preliminaries to a theory of speech disfluencies," Ph.D. dissertation, Univ. California, Berkeley, 1994.
[7] P. A. Heeman and J. F. Allen, "Speech repairs, intonational phrases, and discourse markers: Modeling speaker's utterances in spoken dialogue," *Comput. Ling.*, vol. 25, no. 4, pp. 527–571, 1999.
[8] S. M. Shieber, "Evidence against the context-freeness of natural language," *Ling. Phil.*, vol. 8, no. 3, pp. 333–344, 1985.
[9] H. Matsui, K. Sato, and Y. Sakakibara, "Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot RNA structures," in *Proc. Comput. Ling. Bioinformatics Conf.*, 2004, pp. 290–299.
[10] M. Honal and T. Schultz, "Correction of disfluencies in spontaneous speech using a noisy-channel approach," in *Proc. Eurospeech*, 2003, pp. 2781–2784.

[11] K. Hall and M. Johnson, "Language modeling using efficient best-first bottom-up parsing," in *Proc. Autom. Speech Recognition Understanding Workshop*, 2003, pp. 507–512.
[12] E. Charniak, K. Knight, and K. Yamada. Syntax-based language models for statistical machine translation. presented at Mach. Translation Summit. [Online] Available: http://www.amtaweb.org/summit/MT-Summit/papers.html
[13] S. M. Shieber and Y. Schabes, "Synchronous tree-adjoining grammars," in *Proc. 13th Int. Conf. Comput. Ling.*, 1990, pp. 253–258.
[14] G. Kikui and T. Morimoto, "Similarity-based identification of repairs in japanese spoken language," in *Proc. 3rd Int. Conf. Spoken Lang. Process.*, 1994, pp. 915–918.
[15] B. Santorini. (1990) Part-of-speech tagging guidlines for the Penn Treebank Project (3rd Rev.). Univ. Pennsylvania, Philadelphia. [Online]. Available: http://www.cis.upenn.edu/~treebank/home.html
[16] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Center Research Comput. Technol., Harvard Univ., Cambridge, MA, Tech. Rep. TR-10-98, 1998.
[17] E. Charniak, "Immediate-head parsing for language models," in *Proc. 39th Annu. Meeting Assoc. Comput. Ling.*, 2001, pp. 116–123.
[18] E. Charniak and M. Johnson, "Coarse-to-fine n-best parsing and maxent discriminative reranking," in *Proc. 43rd Annu. Meeting Assoc. Comput. Ling.*, 2005, pp. 173–180.
[19] D. Marcu, *The Theory and Practice of Discourse Parsing and Summarization*. Cambridge, MA: MIT Press, 2000.
[20] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Comput. Ling.*, vol. 26, no. 3, pp. 339–373, 2000.
[21] D. Graff and S. Bird, "Many uses, many annotations for large speech corpora: Switchboard and TDT as case studies," in *Proc. LREC*, 2000, pp. 427–433.
[22] M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer, *Dysfluency Annotation Stylebook for the Switchboard Corpus*. Philadelphia, PA: Linguistic Data Consortium, 1995.
[23] A. Bies, M. Ferguson, K. Katz, and R. MacIntyre, *Bracketting Guideliness for Treebank II Style Penn Treebank Project*. Philadelphia, PA: Linguistic Data Consortium, 1995.
[24] A. Taylor, *Bracketing Switchboard: An Addendum to the Treebank II Bracketing Guidelines*. Philadelphia, PA: Linguistic Data Consortium, 1996.
[25] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, B. Peskin, and M. Harper, "The ICSI-SRI-UW metadata extraction system," in *Proc. Int. Conf. Spoken Lang. Process.*, 2004, pp. 577–580.
[26] D. Engel, E. Charniak, and M. Johnson, "Parsing and disfluency placement," in *Proc. Conf. Empirical Methods in Natural Lang. Process.*, 2002, pp. 49–54.
[27] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," in *Proc. Human Lang. Technol. North Amer. Chap. Assoc. Comp. Linguistics (HLT-NAACL)*, 2006.



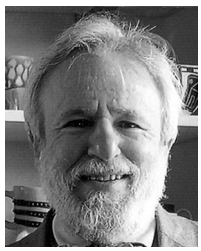
Matthew Lease (S'06) received the B.Sc. degree from the University of Washington, Seattle, in 1999, attending as a Washington Scholar, and the M.Sc. degree from Brown University, Providence, RI, in 2004. He is currently pursuing the Ph.D. degree at Brown University.

His work in natural language processing has focused on improving automated techniques for understanding spontaneous speech, and based on his work at a 2005 Johns Hopkins University workshop, he was recently awarded National Science Foundation support for further study of the interrelation of parsing and disfluency modeling. He has also worked in medical informatics, image compression, and ubiquitous computing, and he has a pseudo-Erdös number of 2 thanks to Vance Faber.

Mark Johnson received the B.Sc. (hons) degree in theoretical chemistry from the University of Sydney, Sydney, Australia, in 1979, the M.A. degree in linguistics from the University of California at San Diego, La Jolla, in 1984, and the Ph.D. degree in linguistics from Stanford University, Stanford, CA, in 1987.

After a Postdoctoral Fellowship at the Massachusetts Institute of Technology, Cambridge, he became an Assistant Professor in the Department of Cognitive and Linguistic Sciences, Brown University, Providence, RI, in 1988, where he is currently Professor of cognitive and linguistic sciences and computer science. His primary research topic is computational linguistics, specializing in statistical models of language and their application to text and speech understanding.

Dr. Johnson is a member of the Association for Computational Linguistics and was its President in 2003.



Eugene Charniak received the A.B. degree in physics from University of Chicago, Chicago, IL, and the Ph.D. degree in computer science from the Massachusetts Institute of Technology, Cambridge.

He is Professor of Computer science and cognitive science at Brown University and past Chair of the Department of Computer Science. He has published four books, the most recent being *Statistical Language Learning* (MIT Press, 1996). His research has always been in the area of language understanding or technologies which relate to it. Over the last 15 years, he has been interested in statistical techniques for parsing, speech recognition, and other areas of language processing.

Dr. Charniak is a Fellow of the American Association of Artificial Intelligence and was previously a Councillor of the organization.