A TAG-based noisy channel model of speech repairs

Mark Johnson
Brown University
Providence, RI 02912
mj@cs.brown.edu

Eugene Charniak
Brown University
Providence, RI 02912
ec@cs.brown.edu

Abstract

This paper describes a noisy channel model of speech repairs, which can identify and correct repairs in speech transcripts. A syntactic parser is used as the source model, and a novel type of TAG-based transducer is the channel model. The use of TAG is motivated by the intuition that the reparandum is a "rough copy" of the repair. The model is trained and tested on the Switchboard disfluency-annotated corpus.

1 Introduction

Most spontaneous speech contains disfluencies such as partial words, filled pauses (e.g., "uh", "um", "huh"), explicit editing terms (e.g., "I mean"), parenthetical asides and repairs. Of these repairs pose particularly difficult problems for parsing and related NLP tasks. This paper presents an explicit generative model of speech repairs and shows how it can eliminate this kind of disfluency.

While speech repairs have been studied by psycholinguists for some time, as far as we know this is the first time a probabilistic model of speech repairs based on a model of syntactic structure has been described in the literature. Probabilistic models have the advantage over other kinds of models that they can in principle be integrated with other probabilistic models to produce a combined model that uses all available evidence to select the globally optimal analysis. Shriberg and Stolcke (1998) studied the location and distribution of repairs in the Switchboard corpus, but did not propose an actual model of repairs. Heeman and Allen (1999) describe a noisy channel model of speech repairs, but leave "extending the model to incorporate higher level syntactic ... processing" to future work. The previous work most closely related to the current work is Charniak and Johnson (2001), who used a boosted decision stub classifier to classify words as edited or not on a word by word basis, but do not identify or assign a probability to a repair as a whole.

There are two innovations in this paper. First, we demonstrate that using a syntactic parser-based language model Charniak (2001) instead of bi/trigram language models significantly improves the accuracy of repair detection and correction. Second, we show how Tree Adjoining Grammars (TAGs) can be used to provide a precise formal description and probabilistic model of the crossed dependencies occurring in speech repairs.

The rest of this paper is structured as follows. The next section describes the noisy channel model of speech repairs and the section after that explains how it can be applied to detect and repair speech repairs. Section 4 evaluates this model on the Penn 3 disfluency-tagged Switchboard corpus, and section 5 concludes and discusses future work.

2 A noisy channel model of repairs

We follow Shriberg (1994) and most other work on speech repairs by dividing a repair into three parts: the *reparandum* (the material repaired), the *interregnum* that is typically either empty or consists of a filler, and the *repair*. Figure 1 shows these three parts for a typical repair.

Most current probabilistic language models are based on HMMs or PCFGs, which induce linear or tree-structured dependencies between words. The relationship between reparandum and repair seems to be quite different: the repair is a "rough copy" of the reparandum, often incorporating the same or very similar words in roughly the same word order. That is, they seem to involve "crossed" dependencies between the reparandum and the repair, shown in Figure 1. Languages with an unbounded number of crossed dependencies cannot be described by a context-free or finite-state grammar, and crossed dependencies like these have been used to argue natural languages

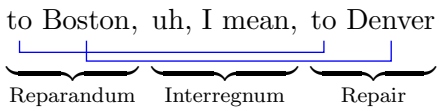

Figure 1: The structure of a typical repair, with crossing dependencies between reparandum and repair.

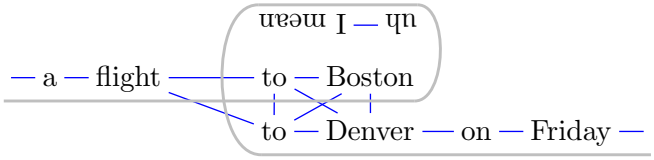

Figure 2: The "helical" dependency structure induced by the generative model of speech repairs for the repair depicted in Figure 1.

are not context-free Shieber (1985). Mildly context-sensitive grammars, such as Tree Adjoining Grammars (TAGs) and Combinatory Categorial Grammars, can describe such crossing dependencies, and that is why TAGs are used here.

Figure 2 shows the combined model's dependency structure for the repair of Figure 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 genome models, and in fact TAGs are being used to model genomes for very similar reasons.

The noisy channel model described here involves two components. A *language model* defines a probability distribution $\mathrm{P}(X)$ over the source sentences X, which do not contain repairs. The *channel model* defines a conditional probability distribution $\mathrm{P}(Y|X)$ of surface sentences Y, which may contain repairs, given source sentences. In the work reported here, X is a word string and Y is a speech transcription not containing punctuation or partial words. We use two language models here: a bigram language model, which is used in the search process, and a syntactic parser-based language model Charniak (2001), which is used to rescore a set of the most likely analysis obtained using the bigram model. Because the language model is responsible for generating the well-formed sentence X, it is reasonable to expect that a language model that can model more global properties of sentences will lead to better performance, and the results presented here show that this is the case. The channel model is a stochastic TAG-based transducer; it is responsible for generating the repairs in the transcript Y, and it uses the ability of TAGs to straightforwardly model crossed dependencies.

2.1 Informal description

Given an observed sentence Y we wish to find the most likely source sentence $\widehat{X}$, where:

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

This is the same general setup that is used in statistical speech recognition and machine translation, and in these applications syntax-based language models $\mathrm{P}(Y)$ yield state-of-the-art performance, so we use one such model here.

The channel model $\mathrm{P}(Y|X)$ generates sentences Y given a source X. A repair can potentially begin before any word of X. When a repair has begun, the channel model incrementally processes the succeeding words from the start of the repair. Before each succeeding word either the repair can end or else a sequence of words can be inserted in the reparandum. At the end of each repair, a (possibly null) interregnum is appended to the reparandum.

The intuition motivating the channel model design is that the words inserted into the reparandum are very closely related those in the repair. Indeed, in our training data over 60% of the words in the reparandum are exact copies of words in the repair; this similarity is strong evidence of a repair. The channel model is designed so that exact copy reparandum words will have high probability.

We assume that X is a substring of Y, i.e., that the source sentence can be obtained by deleting words from Y, so for a fixed observed sentence there are only a finite number of possible source sentences. However, the number of source sentences grows exponentially with the length of Y, so exhaustive search is probably infeasible.

TAGs provide a systematic way of formalizing the channel model, and their polynomial-time dynamic programming parsing algorithms can be used to search for likely repairs, at least when used with simple language models like a bigram language model. In this paper we first identify the 20 most likely analysis of each sentence using the TAG channel model together with a bigram language model. Then each of these analysis is rescored using the TAG channel model and a syntactic parser based language model.

The TAG channel model's analysis do not reflect the syntactic structure of the sentence being analyzed; instead they encode the crossed dependencies of the speech repairs. If we want to use TAG dynamic programming algorithms to efficiently search for repairs, 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; this motivates our use of a bigram language model.

On the other hand, it seems desirable to use a language model that is sensitive to more global properties of the sentence, and we do this by reranking the initial analysis, replacing the bigram language model with a syntactic parser based model. We do not need to intersect this parser based language model with our TAG channel model since we evaluate each analysis separately.

2.2 The TAG channel model

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 Shieber and Schabes (1990), 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* corresponds to 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*.

Figure 3 shows the TAG rules used to generate this example. The nonterminals in this grammar are of the form N_{w_x}, $\mathrm{R}_{w_y:w_x}$ 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 $\mathrm{R}_{w_y:w_x}$ that the preceding words w_y and w_x were part of a repair. The nonterminal I generates words in the interregnum of a repair. Encoding the preceding words in the TAGs nonterminals permits the channel model to be sensitive to lexical properties of the preceding words. The start symbol is $\mathrm{N}_{\$}$, where '$' is a distinguished symbol used to indicate the beginning and end of sentences.

2.3 Estimating the repair channel model from data

The model is trained from the disfluency and POS tagged Switchboard corpus on the LDC Penn tree bank III CD-ROM (specifically, the files under `dysfl/dps/swbd`). This version of the corpus annotates the beginning and ending positions of repairs as well as fillers, editing terms, asides, etc., which might serve as the interregnum in a repair. The corpus also includes punctuation and partial words, which are ignored in both training and evaluation here since we felt that in realistic applications these would not be available in speech recognizer output. The transcript of the example of Figure 1 would look something like the following:

```
a/DT flight/NN [to/IN Boston/NNP +
{F uh/UH} {E I/PRP mean/VBP} to/IN
Denver/NNP] on/IN Friday/NNP
```

In this transcription the repair is the string from the opening bracket "`[`" to the interruption point "`+`"; the interregnum is the sequence of braced strings following the interregnum, and the repair is the string that begins at the end of the interregnum and ends at the closing bracket "`]`". The interregnum consists of the braced

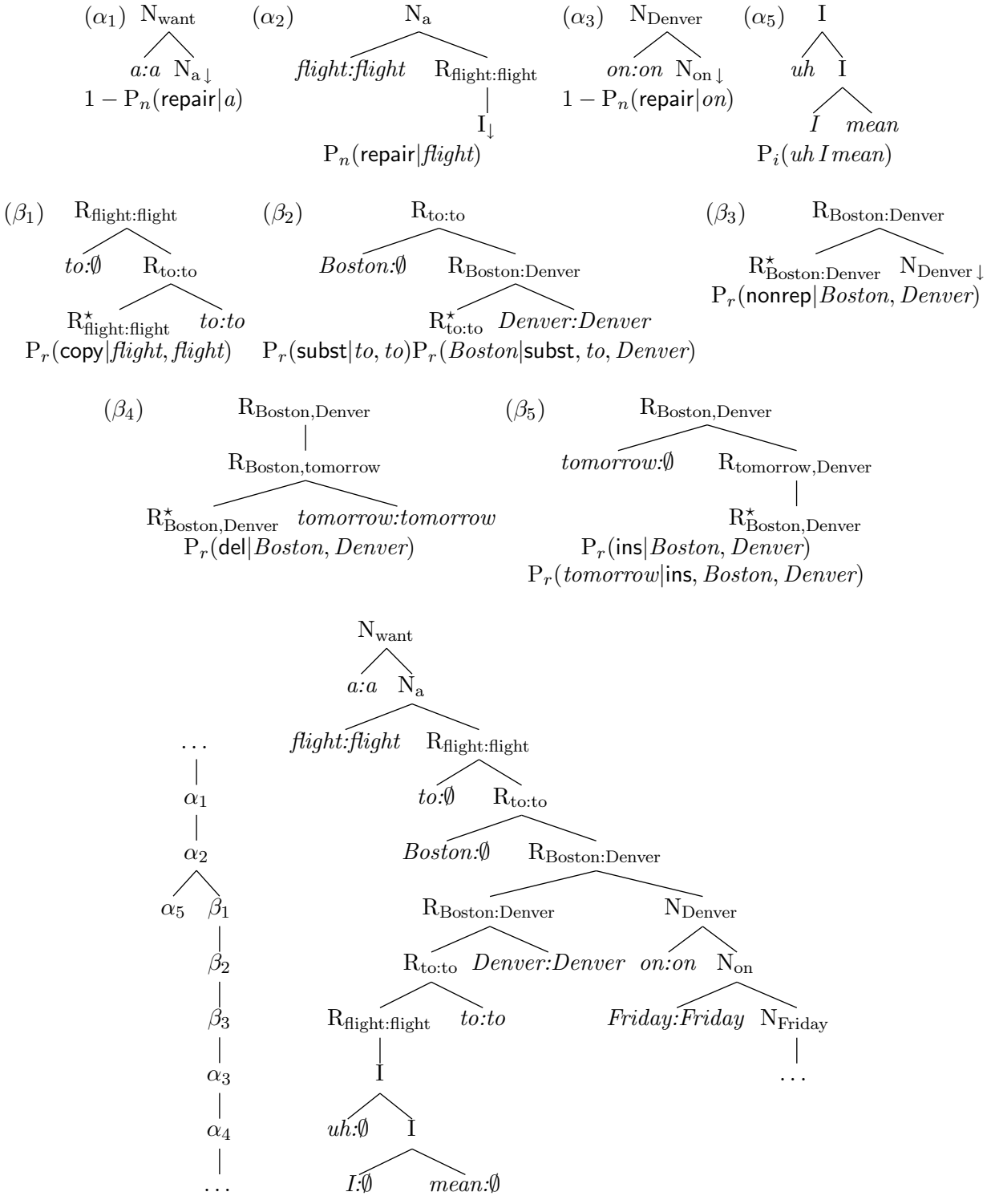

Figure 3: The TAG rules used to generate the example shown in Figure 1 and their respective weights, and the corresponding derivation and derived trees.

expressions immediately following the interruption point. We used the disfluency tagged version of the corpus for training rather than the parsed version because the parsed version does not mark the interregnum, but we need this information for training our repair channel model. Testing was performed using data from the parsed version since this data is cleaner, and

it enables a direct comparison with earlier work. We followed Charniak and Johnson (2001) and split the corpus into main training data, held-out training data and test data as follows: main training consisted of all sw[23]*.dps files, held-out training consisted of all sw4[5-9]*.dps files and test consisted of all sw4[0-1]*.mrg files.

We now describe how the weights on the TAG productions described in subsection 2.2 are estimated from this training data. In order to estimate these weights we need to know the TAG derivation of each sentence in the training data. In order to uniquely determine this we need the not just the locations of each reparandum, interregnum and repair (which are annotated in the corpus) but also the crossing dependencies between the reparandum and repair words, as indicated in Figure 1.

We obtain these by aligning the reparandum and repair strings of each repair using a minimum-edit distance string aligner with the following alignment costs: aligning identical words costs 0, aligning words with the same POS tag costs 2, an insertion or a deletion costs 4, aligning words with POS tags that begin with the same letter costs 5, and an arbitrary substitution costs 7. 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 POS 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 evaluate the quality of the alignments since they are not in themselves the object of this exercise, they seem to be fairly good.

From our training data we estimate a number of conditional probability distributions. These estimated probability distributions are the linear interpolation of the corresponding empirical distributions from the main sub-corpus using various subsets of conditioning variables (e.g., bigram models are mixed with unigram models, etc.) using Chen's bucketing scheme Chen and Goodman (1998). As is commonly done in language modelling, the interpolation coefficients are determined by maximizing the likelihood of the held out data counts using EM. Special care was taken to ensure that all distributions over words ranged over (and assigned non-zero 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.

The first distribution is defined over the words in source sentences (i.e., that do not contain reparandums or interregnums). $\mathrm{P}_n(\mathsf{repair}|W)$ is the probability of a repair beginning after a word W in the source sentence X; it is estimated from the training sentences with reparandums and interregnums removed. Here and in what follows, W ranges over $\Sigma \cup \{\$\}$, where '$' is a distinguished beginning-of-sentence marker. For example, $\mathrm{P}_n(\mathsf{repair}|\textit{flight})$ is the probability of a repair beginning after the word *flight*. Note that repairs are relatively rare; in our training data $\mathrm{P}_n(\mathsf{repair}) \approx 0.02$, which is a fairly strong bias against repairs.

The other distributions are defined over aligned reparandum/repair strings, and are estimated from the aligned repairs extracted from the training data. In training we ignored all overlapping repairs (i.e., cases where the reparandum of one repair is the repair of another). (Naturally, in testing we have no such freedom.) We analyze each repair as consisting of n aligned word pairs (we describe the interregnum model later). M_i is the ith reparandum word and R_i is the corresponding repair word, so both of these range over $\Sigma \cup \{\emptyset\}$. We define M_0 and R_0 to be source sentence word that preceded the repair (which is '$' if the repair begins at the beginning of a sentence). We define M'_i and R'_i to be the last non-$\emptyset$ reparandum and repair words respectively, i.e., $M'_i = M_i$ if $M_i \neq \emptyset$ and $M'_i = M'_{i-1}$ otherwise. Finally, $T_i, i = 1 \ldots n+1$, which indicates the type of repair that occurs at position i, ranges over $\{\mathsf{copy}, \mathsf{subst}, \mathsf{ins}, \mathsf{del}, \mathsf{nonrep}\}$, where $T_{n+1} = \mathsf{nonrep}$ (indicating that the repair has ended), and for $i = 1 \ldots n$, $T_i = \mathsf{copy}$ if $M_i = R_i$, $T_i = \mathsf{ins}$ if $R_i = \emptyset$, $T_i = \mathsf{del}$ if $M_i = \emptyset$ and $T_i = \mathsf{subst}$ otherwise.

The distributions we estimate from the aligned repair data are the following.

$\mathrm{P}_r(T_i|M'_{i-1}, R'_{i-1})$ is the probability of seeing repair type T_i following the reparandum word M'_{i-1} and repair word R'_{i-1}; e.g., $\mathrm{P}_r(\mathsf{nonrep}|\textit{Boston}, \textit{Denver})$ is the probability of the repair ending when *Boston* is the last reparandum word and *Denver* is the last repair word.

$\mathrm{P}_r(M_i|T_i = \mathsf{ins}, M'_{i-1}, R'_i)$ is the probability that M_i is the word that is inserted into the reparandum (i.e., $R_i = \emptyset$) given that some word

is substituted, and that the preceding reparandum and repair words are M'_{i-1} and R'_i. For example $\mathrm{P}_r(\textit{tomorrow}|\mathsf{ins}, \textit{Boston}, \textit{Denver})$ is the probability that the word *tomorrow* is inserted into the reparandum after the words *Boston* and *Denver*, given that some word is inserted.

$\mathrm{P}_r(M_i|T_i = \mathsf{subst}, M'_{i-1}, R'_i)$ is the probability that M_i is the word that is substituted in the reparandum for R'_i, given that some word is substituted. For example, $\mathrm{P}_r(\textit{Boston}|\mathsf{subst}, \textit{to}, \textit{Denver})$ is the probability that *Boston* is substituted for *Denver*, given that some word is substituted.

Finally, we also estimated a probability distribution $\mathrm{P}_i(W)$ over interregnum strings as follows. Our training corpus annotates what we call interregnum expressions, such as *uh* and *I mean*. We estimated a simple unigram distribution over all of the interregnum expressions observed in our training corpus, and also extracted the empirical distribution of the number of interregnum expressions in each repair. Interregnums are generated as follows. First, the number k of interregnum expressions is chosen using the empirical distribution. Then k interregnum expressions are independently generated from the unigram distribution of interregnum expressions, and appended to yield the interregnum string W.

The weighted TAG that constitutes the channel model is straight forward to define using these conditional probability distributions. Note that the language model generates the source string X. Thus the weights of the TAG rules condition on the words in X, but do not generate them.

There are three different schema defining the initial trees of the TAG. These correspond to analyzing a source word as not beginning a repair (e.g., α_1 and α_3 in Figure 3), analyzing a source word as beginning a repair (e.g., α_2), and generating an interregnum (e.g., α_5).

Auxiliary trees generate the paired reparandum/repair words of a repair. There are five different schema defining the auxiliary trees corresponding to the five different values that T_i can take. Note that the nonterminal $R_{m,r}$ expanded by the auxiliary trees is annotated with the last reparandum and repair words M'_{i-1} and R'_{i-1} respectively, which makes it possible to condition the rule's weight on these words.

Auxiliary trees of the form (β_1) generate reparandum words that are copies of the corresponding repair words; the weight on such trees is $\mathrm{P}_r(\mathsf{copy}|M'_{i-1}, R'_{i-1})$. Trees of the form ($\beta_2$) substitute a reparandum word for a repair word; their weight is $\mathrm{P}_r(\mathsf{subst}|M'_{i-1}, R'_{i-1})\mathrm{P}_r(M_i|\mathsf{subst}, M'_{i-1}, R'_i)$. Trees of the form ($\beta_3$) end a repair; their weight is $\mathrm{P}_r(\mathsf{nonrep}|, M'_{i-1}, R'_{i-1})$. Auxiliary trees of the form (β_3) end a repair; they are weighted $\mathrm{P}_r(\mathsf{nonrep}|M'_{i-1}, R'_{i-1})$. Auxiliary trees of the form (β_4) permit the repair word R'_{i-1} to be deleted in the reparandum; the weight of such a tree is $\mathrm{P}_r(\mathsf{del}|M'_{i-1}, R'_{i-1})$. Finally, auxiliary trees of the form (β_5) generate a reparandum word M_i is inserted; the weight of such a tree is $\mathrm{P}_r(\mathsf{ins}|M'_{i-1}, R'_{i-1})\mathrm{P}_r(M_i|\mathsf{ins}, M'_{i-1}, R'_{i-1})$.

3 Detecting and repairing speech repairs

The TAG just described is not probabilistic; informally, it does not include the probability costs for generating the source words. However, it is easy to modify the TAG so it does include a bigram model that does generate the source words, since each nonterminal encodes the preceding source word. That is, we multiply the weights of each TAG production given earlier that introduces a source word R_i by $\mathrm{P}_n(R_i|R_{i-1})$. The resulting stochastic TAG is in fact exactly the intersection of the channel model TAG with a bigram language model.

The standard n^5 bottom-up dynamic programming parsing algorithm can be used with this stochastic TAG. Each different parse of the observed string Y with this grammar corresponds to a way of analyzing Y in terms of a hypothetical underlying sentence X and a number of different repairs. In our experiments below we extract the 20 most likely parses for each sentence. Since the weighted grammar just given does not generate the source string X, the score of the parse using the weighted TAG is $\mathrm{P}(Y|X)$. This score multiplied by the probability $\mathrm{P}(X)$ of the source string using the syntactic parser based language model, is our best estimate of the probability of an analysis.

However, there is one additional complication that makes a marked improvement to the model's performance. Recall that we use the standard bottom-up dynamic programming TAG parsing algorithm to search for candidate parses. This algorithm has n^5 running time, where n is the length of the string. 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 non-repair rules, or equivalently, the odds that this substring constitutes a repair. The substrings with high repair 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 to compute global analyses of the entire sentence. Second, there are rare cases in which the same substring functions as both repair and reparandum (i.e., the repair string is itself repaired again). 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 a repair), 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).

4 Evaluation

This section describes how we evaluate our noisy model. As mentioned earlier, following Charniak and Johnson (2001) our test data consisted of all Penn III Switchboard tree-bank sw4[0-1]*.mrg files. However, our test data differs from theirs in that in this test we deleted all partial words and punctuation from the data, as this results in a more realistic test situation.

Since the immediate goal of this work is to produce a program that identifies the words of a sentence that belong to the reparandum of a repair construction (to a first approximation these words can be ignored in later processing), our evaluation focuses on the model's performance in recovering the words in a reparandum. That is, the model is used to classify each word in the sentence as belonging to a reparandum or not, and all other additional structure produced by the model is ignored.

We measure model performance using standard precision p, recall r and f-score f, measures. If n_c is the number of reparandum words the model correctly classified, n_t is the number of true reparandum words given by the manual annotations and n_m is the number of words the model predicts to be reparandum words, then the precision is n_c/n_m, recall is n_c/n_t, and f is $2pr/(p+r)$.

For comparison we include the results of running the word-by-word classifier described in Charniak and Johnson (2001), but where partial words and punctuation have been removed from the training and test data. We also provide results for our noisy channel model using a bigram language model and a second trigram model where the twenty most likely analyses are rescored. Finally we show the results using the parser language model.

	CJ01′	Bigram	Trigram	Parser
Precision	0.951	0.776	0.774	0.820
Recall	0.631	0.736	0.763	0.778
F-score	0.759	0.756	0.768	0.797

The noisy channel model using a bigram language model does a slightly worse job at identifying reparandum and interregnum words than the classifier proposed in Charniak and Johnson (2001). Replacing the bigram language model with a trigram model helps slightly, and parser-based language model results in a significant performance improvement over all of the others.

5 Conclusion and further work

This paper has proposed a novel noisy channel model of speech repairs and has used it to identify reparandum words. One of the advantages of probabilistic models is that they can be integrated with other probabilistic models in a principled way, and it would be interesting to investigate how to integrate this kind of model of speech repairs with probabilistic speech recognizers.

There are other kinds of joint models of reparandum and repair that may produce a better reparandum detection system. We have experimented with versions of the models described above based on POS bi-tag dependencies rather than word bigram dependencies, but with results very close to those presented here. Still, more sophisticated models may yield better performance.

It would also be interesting to combine this probabilistic model of speech repairs with the word classifier approach of Charniak and Johnson (2001). That approach may do so well because many speech repairs are very short, involving only one or two words Shriberg and

Stolcke (1998), so the reparandum, interregnum and repair are all contained in the surrounding word window used as features by the classifier. On the other hand, the probabilistic model of repairs explored here seems to be most successful in identifying long repairs in which the reparandum and repair are similar enough to be unlikely to have been generated independently. Since the two approaches seem to have different strengths, a combined model may outperform both of them.

References

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

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. The Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Center for Research in Computing Technology, Harvard University.

Peter A. Heeman and James F. Allen. 1999. Speech repairs, intonational phrases, and discourse markers: Modeling speaker's utterances in spoken dialogue. *Computational Linguistics*, 25(4):527–571.

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING 1990)*, pages 253–258.

Stuart M. Shieber. 1985. Evidence against the Context-Freeness of natural language. *Linguistics and Philosophy*, 8(3):333–344.

Elizabeth Shriberg and Andreas Stolcke. 1998. How far do speakers back up in repairs? a quantitative model. In *Proceedings of the International Conference on Spoken Language Processing*, volume 5, pages 2183–2186, Sydney, Australia.

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