Word Buffering Models for Improved Speech Repair Parsing*

Tim Miller
University of Minnesota – Twin Cities
tmill@cs.umn.edu

Abstract

This paper describes a time-series model for parsing transcribed speech containing disfluencies. This model differs from previous parsers in its explicit modeling of a buffer of recent words, which allows it to recognize repairs more easily due to the frequent overlap in words between errors and their repairs. The parser implementing this model is evaluated on the standard Switchboard transcribed speech parsing task for overall parsing accuracy and edited word detection.

1 Introduction

Speech repair is a phenomenon in spontaneous speech where a speaker interrupts the flow of speech (at what's called the *interruption point*), backtracks some number of words (the *reparandum*), and continues the utterance with material meant to replace the reparandum (the *alteration*).[1] The utterance can be rendered syntactically correct by excising all the words that the speaker skipped over when backtracking. Speech with repair is difficult for machines to process because in addition to detecting repair, a system must know what words are meant to be excised, and parsing systems must determine how to form a grammatical structure out of the set of words comprising both the error speech and the correct speech.

Recent approaches to syntactic modeling of speech with repairs have shown that significant gains in parsing accuracy can be achieved by modeling the syntax of repairs (Hale et al., 2006; Core and Schubert, 1999). In addition, others have shown that a parser based on a time-series model that explicitly represents the incomplete constituents in fluent and disfluent speech can also improve parsing accuracy (Miller and Schuler, 2008). However, these parsing approaches are still not as accurate at detecting reparanda as classification systems which use a variety of features to detect repairs (Charniak and Johnson, 2001; Johnson and Charniak, 2004; Heeman and Allen, 1999).

One highly salient feature which classification systems use to detect repair is the repetition of words between the error and the repair. Johnson and Charniak report that 60% of words in the alterations are copies of words in reparanda in the Switchboard corpus. Typically, this information is not available to a parser trained on context-free grammars.

Meanwhile, psycholinguistic models suggest that the human language system makes use of buffers both to keep track of recent input (Baddeley et al., 1998) and to smooth out generation (Levelt, 1989). These buffers are hypothesized to contain representations of recent phonological events, suggesting that there is a short window where new input might be compared to recent input. This could be represented as a buffer which predicts or detects repeated input in certain constrained circumstances.

This paper describes a hybrid parsing system operating on transcribed speech which combines an incremental parser implemented as a probabilistic time-series model, as in Miller and Schuler, with a buffer of recent words meant to loosely model something like a phonological loop, which should better account for word repetition effects in speech repair.

2 Background

This work uses the Switchboard corpus (Godfrey et al., 1992) for both training and testing. This corpus contains transcribed and syntactically annotated conversations between human interlocutors. The reparanda in speech repairs are ulti-

*This research was supported by NSF CAREER award 0447685. The views expressed are not necessarily endorsed by the sponsors .

[1]This terminology follows Shriberg (1994).

mately dominated by the EDITED label, and in cases where the reparandum ends with an unfinished constituent, the lowest constituent label is augmented with the -UNF tag. These annotations provide necessary but not sufficient information for parsing speech with repairs, and thus many improvements in performing this task come as the result of modifying these annotations in the training data.

As mentioned above, both Hale and colleagues (2006) and Miller and Schuler (2008) showed that speech repairs contain syntactic regularities, which can improve the parsing of transcribed speech with repairs when modeled properly. Hale et al. used 'daughter annotation', which adds the label of an EDITED node's child to the EDITED label itself, and '-UNF propagation', which labels every node between an original -UNF node and the EDITED with an -UNF tag. Miller and Schuler used a 'right-corner transform' to convert standard phrase structure trees of the Penn Treebank into 'right-corner trees', which have highly left-branching structure and non-standard tree categories representing incomplete constituents being recognized. These trees can be mapped into a fixed-depth Hierarchical Hidden Markov Model to achieve improved parsing and reparandum-finding results over standard CYK parsers.

Work by Johnson and Charniak (2004; 2001) uses much of the same structure, but is not a parsing approach per se. In earlier work, they used a boosting algorithm using word identity and category features to classify individual words as part of a reparandum or not, and achieved very impressive accuracy. More recent work uses a tree-adjoining grammar (TAG) to model the overlap in words and part-of-speech tags between reparandum and alteration as context sensitive syntax trees. A parser is then used to rank the multiple outputs of the TAG model with reparandum words removed.

Another approach that makes use of the correspondence between words in the reparandum and alteration is Heeman and Allen (1999). This approach uses several sources of evidence, including word and POS correspondence, to predict repair beginnings and correct them (by predicting how far back they are intended to retrace). This model includes random variables between words that correspond to repair state, and in a repair state, allows words in the reparandum to 'license' words in the alteration with high probability, accounting for the high percentage of copied words and POS tags between reparandum and alteration.

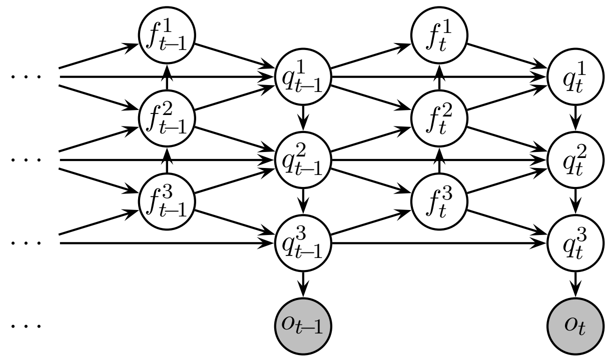


Figure 1: Graphical representation of the dependency structure in a standard Hierarchic Hidden Markov Model with $D = 3$ hidden levels that can be used to parse syntax. Circles denote random variables, and edges denote conditional dependencies. Shaded circles denote variables with observed values.

3 Model Description

This work is based on a standard Hierarchical Hidden Markov Model parser (Schuler, 2009), with the addition of two new random variables for tracking the state of speech repair. The HHMM framework is a desirable starting point for this work for two reasons: First, its definition in terms of a graphical model makes it easy to think about and to add new random variables. Second, the HHMM parser operates incrementally in a left-to-right fashion on word input, which allows this system to run in a single pass, conditioning current words on a hypothesized buffer and interruption point variable. The incremental nature of this system is a constraint that other systems are not bound by, but makes this model more psycholinguistically plausible. In comparison, a CYK parsing framework attempting to use the same probabilistic model of word dependency between reparanda and alterations would need to do a second pass after obtaining the most likely parses, in order to tell if a particular word's generation probability in a specific parse is influenced by a recent repair.

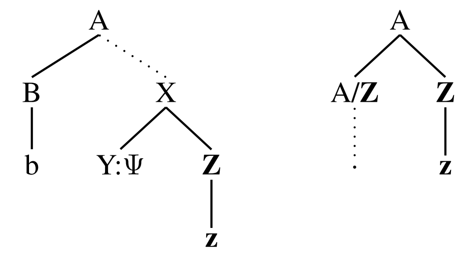
The graphical model representation of this framework is illustrated in Figures 1 and 4. The original model, shown in Figure 1, has complex variables Q and F broken down into several q^d_t and f^d_t for time step t and depth d. These random variables will be explained shortly, but for now suffice it to say that in this work they are unaltered from the original HHMM parsing framework, while those labeled I and B (Figure 4) are additions specific to the system described in this paper. This section will next describe the standard HHMM parsing framework, before describing how this work augments it.

3.1 Right-corner Transform

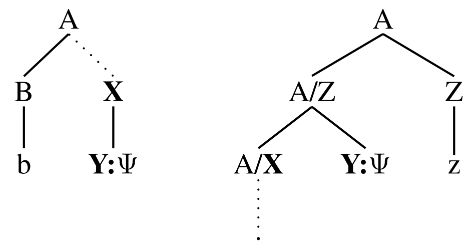
The HHMM parser consists of stacks of a fixed depth, which contain hypotheses of constituents that are being processed. In order to minimize the number of stack levels needed in processing, the phrase structure trees in the training set are modified using a 'right-corner transform', which converts right expansion in trees to left expansion, leaving heavily left-branching structure requiring little depth. The right-corner transform used in this paper is simply the left-right dual of a left-corner transform (Johnson, 1998a).

The right-corner transform can be defined as a recursive algorithm on phrase-structure trees in Chomsky Normal Form (CNF). Trees are converted to CNF first by binarizing using standard linguistically-motivated techniques (Klein and Manning, 2003; Johnson, 1998b). Remaining unbinarized structure is binarized in a brute force fashion, creating right-branching structure by creating a single node which dominates the two rightmost children of a 'super-binary' tree, with the label being the concatenation of its children's labels (see Figure 2).

Taking this CNF phrase structure tree as input, the right-corner transform algorithm keeps track of two separate trees, the original and the new right-corner tree it is building. This process begins at the right-most preterminal of the original tree, and works its way up along the right 'spine', while building its way down a corresponding left spine of the new right-corner tree. The trees below shows the first step of the algorithm, with the tree on the left being disassembled, the tree on the right being built from its parts, and the working positions in the trees shown in bold.



The bottom right corner of the original tree is made the top right corner of the new tree, and the left corner of the new tree is made the new working position and given a 'slash' category A/Z. The 'slash' category label A/Z represents a tree that is the start of a constituent of type A that needs a right-child of type Z in order to complete. The new right-corner of the original tree is the parent (X) of the previous right corner, and its subtree is now added to the right-corner derivation:



After the first step, the subtrees moved over to the right-corner tree may have more complex substructure than a single word (in this case, Ψ represents that possibly complex structure). After being attached to the right-corner tree in the correct place, the algorithm is recursively applied to that now right-branching substructure.

Again, the left child is given a new slash category: The 'active constituent' (the left side of a slash category) is inherited from the root, and the 'awaited constituent' (the right side of a slash category) is taken from the constituent label of the right-corner it came from.

This algorithm proceeds iteratively up the right spine of the original tree, moving structure to the right-corner tree and recursively transforming it as it is added. The final step occurs when the original root (A in this case) is reduced to having a single child, in which case its child is added as a child of the leftmost current branch of the right-corner tree, and it is transformed recursively.

Figures 2 and 3 show an example tree from the Switchboard corpus before and after the right-corner transform is applied.

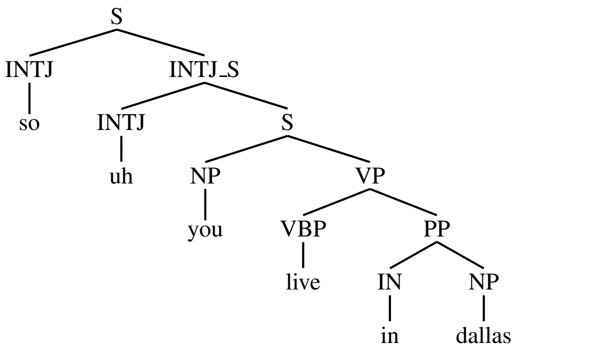


Figure 2: Input to the right-corner transform. This tree also shows an example of the 'brute-force' binarization done on super-binary branches that cannot be otherwise be binarized with linguistically-motivated rules.

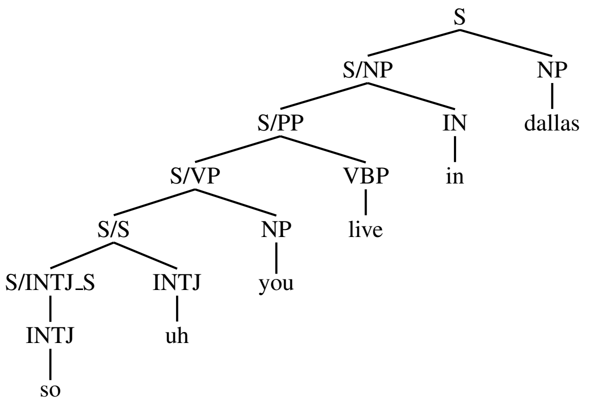


Figure 3: Right-corner transformed version of the tree in Figure 2.

3.2 Hierarchical Hidden Markov Model

A Hierarchical Hidden Markov Model is essentially an HMM with a specific factorization that is useful in many domains — the hidden state at each time step is factored into d random variables which function as a stack, and d additional random variables which regulate the operations of the stack through time. For the model of speech repair presented here, an interruption point is identified by one of these regulator variables firing earlier than it would in fluent speech. This concept will be formalized below. The stack regulating random variables are typically marginalized out when performing inference on a sequence.

While the vertical direction of the hidden sub-states (at a fixed t) represents a stack at a single point in time, the horizontal direction of the hidden sub-states (at a fixed d) can be viewed as a simple HMM at depth d, expanding the state from the HMM above it across multiple time steps and causing the HMM below it to expand its own states. This interpretation will be useful when formally defining the transitions between the stack elements at different time steps below.

Formally, HMMs characterize speech or text as a sequence of hidden states q_t (which may consist of speech sounds, words, and/or other hypothesized syntactic or semantic information), and observed states o_t at corresponding time steps t (typically short, overlapping frames of an audio signal, or words or characters in a text processing application). A most likely sequence of hidden states $\hat{q}_{1..T}$ can then be hypothesized given any sequence of observed states $o_{1..T}$, using Bayes' Law (Equation 2) and Markov independence assumptions (Equation 3) to define a full $\mathsf{P}(q_{1..T} \mid o_{1..T})$ probability as the product of a *Language Model* (Θ_L) prior probability and an *Observation Model* (Θ_O) likelihood probability:

$$\hat{q}_{1..T} = \underset{q_{1..T}}{\mathrm{argmax}}\, \mathsf{P}(q_{1..T} \mid o_{1..T}) \tag{1}$$

$$= \underset{q_{1..T}}{\mathrm{argmax}} \mathsf{P}(q_{1..T}) \cdot \mathsf{P}(o_{1..T} \mid q_{1..T}) \tag{2}$$

$$\overset{\mathrm{def}}{=} \underset{q_{1..T}}{\mathrm{argmax}} \prod_{t=1}^{T} \mathsf{P}_{\Theta_\mathrm{L}}(q_t \mid q_{t\text{-}1}) \cdot \mathsf{P}_{\Theta_\mathrm{O}}(o_t \mid q_t) \tag{3}$$

Language model transitions $\mathsf{P}_{\Theta_\mathrm{L}}(q_t \mid q_{t-1})$ over complex hidden states q_t can be modeled using synchronized levels of stacked-up component HMMs in a Hierarchic Hidden Markov Model (HHMM) (Murphy and Paskin, 2001). HHMM transition probabilities are calculated in two phases: a 'reduce' phase (resulting in an intermediate, marginalized state f_t), in which component HMMs may terminate; and a 'shift' phase (resulting in a modeled state q_t), in which unterminated HMMs transition, and terminated HMMs are re-initialized from their parent HMMs. Variables over intermediate f_t and modeled q_t states are factored into sequences of depth-specific variables — one for each of D levels in the HMM hierarchy:

$$f_t = \langle f_t^1 \dots f_t^D \rangle \tag{4}$$

$$q_t = \langle q_t^1 \dots q_t^D \rangle \tag{5}$$

Transition probabilities are then calculated as a product of transition probabilities at each level, using level-specific 'reduce' Θ_F and 'shift' Θ_Q models:

$$\mathsf{P}_{\Theta_{\mathrm{L}}}(q_t | q_{t-1}) = \sum_{f_t} \mathsf{P}(f_t | q_{t-1}) \cdot \mathsf{P}(q_t | f_t\, q_{t-1}) \tag{6}$$

$$\overset{\text{def}}{=} \sum_{f_t^{1..D}} \prod_{d=1}^{D} \mathsf{P}_{\Theta_{\mathrm{F}}}(f_t^d | \, f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \cdot \mathsf{P}_{\Theta_{\mathrm{Q}}}(q_t^d | f_t^{d+1} f_t^d \;\, q_{t-1}^d q_t^{d-1}) \tag{7}$$

with f_t^{D+1} and q_t^0 defined as constants.

Shift and reduce probabilities are now defined in terms of finitely recursive FSAs with probability distributions over transition, recursive expansion, and final-state status of states at each hierarchy level. In the HHMM used in this paper, each intermediate state variable is a reduction state variable $f_t^d \in G \cup \{\mathbf{0}, \mathbf{1}\}$ (where G is the set of all nonterminal symbols from the original grammar), representing a reduction to the final syntactic state in G, a horizontal transition to a new awaited category, or a top-down transition to a new active category. Each modeled state variable is a syntactic element ($q_t^d \in G \times G$) with an active and awaited category represented with the slash notation.

The intermediate variable f_t^d is probabilistically determined given a reduction at the stack level below, but is deterministically **0** in the case of a non-reduction at the stack level below. [2]

$$\mathsf{P}_{\Theta_{\mathrm{F}}}(f_t^d \mid f_t^{d+1} q_{t-1}^d q_{t-1}^{d-1}) \overset{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G : [f_t^d = \mathbf{0}] \\ \text{if } f_t^{d+1} \in G : \mathsf{P}_{\Theta_{\text{F-Reduce}}}(f_t^d \mid q_{t-1}^d, q_{t-1}^{d-1}) \end{cases} \tag{8}$$

where $f^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$.

Shift probabilities at each level are defined using level-specific transition $\Theta_{\text{Q-T}}$ and expansion $\Theta_{\text{Q-E}}$ models:

$$\mathsf{P}_{\Theta_{\mathrm{Q}}}(q_t^d \mid f_t^{d+1} f_t^d \; q_{t-1}^d q_t^{d-1}) \overset{\text{def}}{=} \begin{cases} \text{if } f_t^{d+1} \notin G, f_t^d \notin G : [q_t^d = q_{t-1}^d] \\ \text{if } f_t^{d+1} \in G, f_t^d \notin G : \mathsf{P}_{\Theta_{\text{Q-T}}}(q_t^d \mid f_t^{d+1} f_t^d q_{t-1}^d q_t^{d-1}) \\ \text{if } f_t^{d+1} \in G, f_t^d \in G : \mathsf{P}_{\Theta_{\text{Q-E}}}(q_t^d \mid q_t^{d-1}) \end{cases} \tag{9}$$

where $f^{D+1} \in G$ and $q_t^0 = \mathbf{ROOT}$. This model is conditioned on final-state switching variables at and immediately below the current HHMM level. If there is no final state immediately below the current level (the first case above), it deterministically copies the current HHMM state forward to the next time step. If there is a final state immediately below the current level (the second case above), it transitions the HHMM state at the current level, according to the distribution $\Theta_{\text{Q-T}}$. And if the state at the current level is final (the third case above), it re-initializes this state given the state at the level above, according to the distribution $\Theta_{\text{Q-E}}$. The overall effect is that higher-level HMMs are allowed to transition only when lower-level HMMs terminate. An HHMM therefore behaves like a probabilistic implementation of a pushdown automaton (or 'shift-reduce' parser) with a finite stack, where the maximum stack depth is equal to the number of levels in the HHMM hierarchy.

All of the probability distributions defined above can be estimated by training on a corpus of right-corner transformed trees, by mapping tree elements onto the random variables in the HHMM and computing conditional probability tables at each random variable. This process is described in more detail in other work (Schuler et al., in press).

3.3 Interruption Point and Word Buffer

This paper expands upon this standard HHMM parsing model by adding two new sub-models to the hidden variables described above, an interruption point (I) variable, and a word buffer (B) . This model is illustrated in Figure 4, which takes Figure 1 as a starting point and adds random variables just mentioned.

Buffers are hypothesized to be used in the human language system to smooth out delivery of speech (Levelt, 1989). In this work, a buffer of that sort is placed between the syntax generating elements and the observed evidence (words). Its role in this model is not to smooth the flow of speech, but to keep a short memory that enables the speaker to conveniently and helpfully restart when a repair is produced. This in turn gives assistance to a listener trying to understand what the speaker is saying, since the listener also has the last few words in memory.

The I variable implements a state machine that keeps track of the repair status at each time point. The domain of this variable is $\{\mathbf{0}, \mathbf{1}, \mathbf{ET}\}$, where $\mathbf{1}$ indicates the first word of an alteration, **ET** indicates editing terms in between reparandum and alteration, and **0** indicating no repair.[3]

[2]Here $[\cdot]$ is an indicator function: $[\phi] = 1$ if ϕ is true, 0 otherwise.

[3]Actually, **0** can occur during an alteration, but in those cases that fact is indicated by the state of the buffer.

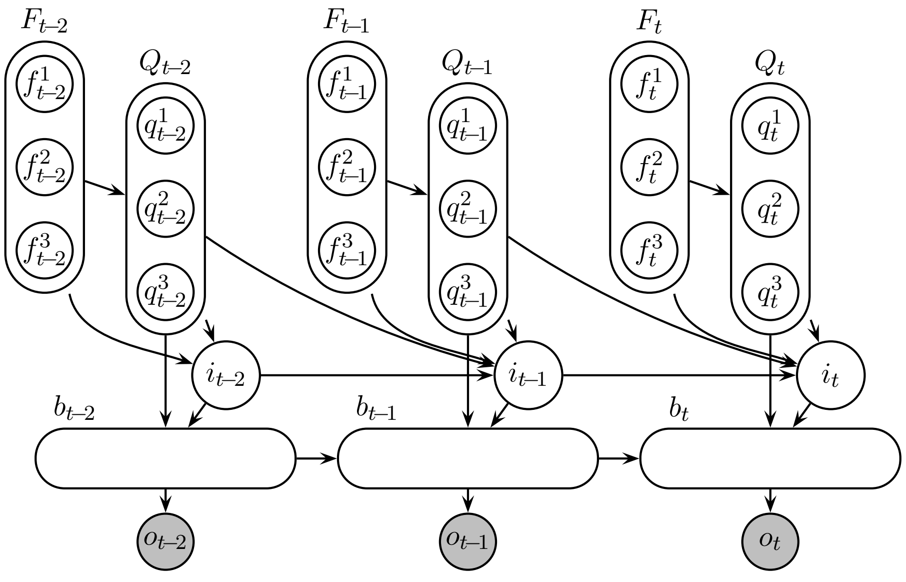


Figure 4: Extended HHMM parsing model with variables for interruption points (I) and a modeled word buffer (B). Arrows within and between complex hidden variables F and Q have been removed for clarity.

The value of I is deterministically constrained in this work by its inputs, but it can be conceived as a conditional probability $\mathsf{P}(i_t \mid i_{t-1}, q_t, q_{t-1}, r_t)$ to allow footholds for future research.[4] While depending formally on many values, in practice its dependencies are highly context-dependent and constrained:

$$\mathsf{P}(i_t \mid i_{t-1}, q_t, q_{t-1}, q_t) \overset{\text{def}}{=}$$
$$\begin{cases} \text{if } i_{t-1} = \mathbf{1} & : [i_t = \mathbf{0}] \\ \text{if } i_{t-1} = \mathbf{ET} \wedge (INTJ \vee PRN) \in q_t & : [i_t = \mathbf{ET}] \\ \text{if } i_{t-1} = \mathbf{ET} & : [i_t = \mathbf{1}] \\ \text{if } i_{t-1} = \mathbf{0} \wedge EDITED \in (q_{t-1} \cup f_t) & \\ \quad \wedge (INTJ \vee PRN) \in q_t & : [i_t = \mathbf{ET}] \\ \text{if } i_{t-1} = \mathbf{0} \wedge EDITED \in (q_{t-1} \cup f_t) & : [i_t = \mathbf{1}] \\ \text{if } i_{t-1} = \mathbf{0} & : [i_t = \mathbf{0}] \end{cases}$$

These conditions are meant to be evaluated in a short-circuiting fashion, i.e., the first condition which is true starting from the top is applied. The default (last) case is most common, going from non-repair to non-repair state. When the syntax generated something with the category EDITED at the last time step (as evidenced by either the modeled state variable q_{t-1} or the reduction state variable f_t depending on the length of the reparandum), the interruption point variable is triggered to change, either to $\mathbf{ET}$ if an interjection (INTJ) or parenthetical (PRN) followed, otherwise to $\mathbf{1}$ for the first word of an alteration. The $\mathbf{ET}$ state continues as long as the syntax at the current level is generating something containing INTJ or PRN.

The random variable for the word buffer is more complex, containing at each time step t an integer index for keeping track of a current position in the buffer ($c_t \in \langle 0, 1, \ldots, n-1 \rangle$ for buffer size n), and an array of several recently generated words ($\vec{w}_t$). This can be represented as the following conditional probability:

$$\mathsf{P}(b_t \mid b_{t-1}, i_t, q_t) = \mathsf{P}(c_t \mid c_{t-1}, i_t) \cdot \mathsf{P}(\vec{w}_t \mid \vec{w}_{t-1}, c_t) \qquad (10)$$

The operation of the buffer is governed by four cases:

Case 1: During normal operation (i.e. for fluent speech), the interruption point variable is $\mathbf{0}$ and at the previous time step the buffer index points at the end of the buffer ($i_t = \mathbf{0} \wedge c_{t-1} = n-1$). In this simple case, the buffer pointer remains pointing at the end position in the buffer ($c_t = n-1$), and the last $n-1$ items in the buffer are deterministically copied backwards one position. A new word is generated probabilistically to occupy the last position in the buffer (where c_t is pointing). This probability is estimated empirically using the same model used in a standard HHMM to generate words, by conditioning the word on the deepest non-empty q_t value in the stack.

Case 2: When an editing term is being generated, ($i_t = \mathbf{ET}$), the buffer is not in use. Practi-

[4]Most obviously, this variable could be made prior to its conditions to be their cause, if a suitable model for the causation of interruption points was designed using prosodic cues. For this work, it is simply an intermediary that is not strictly necessary but makes the model design more intuitive.

cally, this means that the value of the index c and all w^j are just copied over from time $t-1$ to time t. This makes sense psycholinguistically, because a buffer used to smooth speech rates would by definition not be used when speech is interrupted by a repair. It also makes sense from a purely engineering point of view, since words used as editing terms are usually stock phrases and filled pauses that are not likely to have much predictive value for the alteration, and are thus not worth keeping in the buffer. The probability of the actual observed word is modeled the same way word probabilities are modeled in a standard HHMM, conditioned on the deepest non-empty q_t value, and ignoring the buffer.

Case 3: The alteration case applies to the first word after the reparandum and optional editing terms ($i_t = \mathbf{1}$). In this case, the index c_t for the current position of the buffer is obtained by subtracting a number of words to replace, with that number drawn from a prior distribution. This distribution is based on the function $f(k) = 1.22 \cdot 0.45^k$. This function was taken from Shriberg (1996), where it was estimated based on several different training corpora, and provided a remarkable fit to all of them. Since this model uses a fixed size buffer, the values are precomputed and renormalized to form a probability distribution. With a buffer size of only $n = 4$, approximately 96% of the probability mass of the original function is accounted for.

After the indices are computed, the buffer at position c_t is given a word value. The model first decides whether to substitute or copy the previous word over. The probability governing this decision is also determined empirically, by computing how often the first word in a alteration in the Switchboard training set is a copy of the first word it is meant to replace. If the copy operation is selected, the word is added to the buffer without further diluting its probability. If, however, the substitution operation was selected, the word is added to the buffer with probability distributed across all possible words.

Case 4: The final case to account for is alterations of length greater than one ($i_t = \mathbf{0} \wedge c_{t-1} \neq n-1$). This occurs when the current index was moved back more than one position, and so even though i is set to $\mathbf{0}$, the current index into the buffer is not pointing at the end. In this case, again the index c_t is selected according to a prior probability distribution. The value selected from the distribution corresponds to different actions that may be selected when retracing the words in the reparandum to generate the alteration.

The first option is that the current index remains in place, which corresponds to an *insertion* operation, where the alteration is given an extra word relative to the reparandum at its current position. Following an insertion, a new word is generated and placed in the buffer at the current index, with probability conditioned on the syntax at the most recent time step. The second option is to continue the alignment, moving the current index forward one position in the buffer, and then either performing a *substitution* or *copy* operation in alignment with a word from the alteration. Word probabilities for the copy and substitution operations are generated in the same way as for the first word of an alteration. Finally, the current index may skip forward more than one value, performing a *deletion* operation. Deletion skips over words in the reparandum that do not correspond to words in the alteration. After the deletion moves the current index pointer forward, a word is again either copied or substituted against the newly aligned word.

The prior probability distributions over alignment operations is estimated from data in the Switchboard in a similar manner to Johnson and Charniak (2004). Briefly, using the disfluency-annotated section of the Switchboard corpus (.dps files), a list of reparanda and alterations corresponding to one another are compiled. For each pair, the minimal cost alignment is computed, where a copy operation has cost 0, substitution has cost 4, and deletion and insertion each have cost 7. Using these alignments, probabilities are computed using relative frequency counts for both the first word of an alteration, and for subsequent operations. Copy and substitution are the most frequent operations (copying gives information about the repair itself, while substitution can correct the reason for the error), insertion is somewhat less frequent (presumably for specifying further information), and deletion is relatively rare (usually a repair is not made to remove information).

4 Evaluation

This model was evaluated on the Switchboard corpus (Godfrey et al., 1992) of conversational telephone speech between two human interlocu-

System	*Precision*	*Recall*	*F-Score*
Plain CYK	18.01	17.73	17.87
Hale et al. CYK	40.90	35.41	37.96
Hale et al. Lex.	n/a	n/a	*70.0*
TAG	*82.0*	*77.8*	*79.7*
Plain HHMM	43.90	47.36	45.57
HHMM-Back	**44.12**	**57.49**	**49.93**
HHMM-Retrace	**48.82**	**59.41**	**53.59**

Table 1: Table of results of edit-finding accuracy. Italics indicate reported, rather than reproduced, results.

System Configuration	*Parseval-F*	*Edited-F*
Plain CYK	71.03	17.9
Hale et al. CYK	68.47	37.96
Hale et al. Lex.	*80.16*	*70.0*
Plain HHMM	74.23	45.57
HHMM-Back	**74.58**	**49.93**
HHMM-Retrace	**74.23**	**53.59**

Table 2: Table of parsing results.

tors. The input to this system is the gold standard word transcriptions, segmented into individual utterances. The standard train/test breakdown was used, with sections 2 and 3 used for training, and subsections 0 and 1 of section 4 used for testing. Several held-out sentences from the end of section 4 were used during development.

For training, the data set was first standardized by removing punctuation, empty categories, typos, all categories representing repair structure, and partial words – anything that would be difficult or impossible to obtain reliably with a speech recognizer.

The two metrics used here are the standard Parseval F-measure, and Edit-finding F. The first takes the F-score of labeled precision and recall of the non-terminals in a hypothesized tree relative to the gold standard tree. The second measure marks words in the gold standard as edited if they are dominated by a node labeled EDITED, and measures the F-score of the hypothesized edited words relative to the gold standard.

Results are shown in Tables 1 and 2. Table 1 shows detailed results on edited word finding, with two test systems and several related approaches.

The first two lines show results from a re-implementation of Hale et al. parsers. In both those cases, gold standard part-of-speech (POS) tags were supplied to the parser. The following two lines are reported results of a lexicalized parser from Hale et al. and the TAG system of Johnson and Charniak. The final three lines are evaluations of HHMM systems. The first is an implementation of Miller and Schuler, run without gold standard POS tags as input. The second HHMM result is a system much like that described in this paper, but designed to approximate the best result that can come from simply trying to match the first word of an alteration with a recent word. Levelt (1989) notes that in over 90% of repairs, the first word of the alteration is either identical or a member of the same category as the first word of the reparandum, and this clue is enough for listeners to understand what the alteration is meant to replace. This implementation keeps the I variable to model repair state, but rather than a modeled buffer being part of the hidden state, it keeps an observed buffer that simply tracks the last n words seen ($n = 4$ in this experiment). This buffer is used only to generate the first word of a repair, and only when the syntactic state allows the word. Finally, the system described in Section 3 is shown on the final line.

Table 2 shows overall parsing accuracy results, with the same set of systems, with the exception of the TAG system which did not report parsing results.

5 Discussion and Conclusion

These results first show that the main contribution of this paper, a model for a buffer of recent words which influences speech repairs, results in drastic improvements in the ability of an HHMM system to discover edited words. This model does this in a single pass through the observed words, incrementally forming hypotheses about the state of the syntactic process as well as the state of repair, just as humans must recognize spontaneous speech.

Another interesting result is the relative effectiveness of a buffer that is not modeled, but rather just a collection of words used to condition the first words of repair ('HHMM-Back'). While this result is superior to the plain HHMM system, it still falls well short of the retracing model using a modeled buffer. This suggests that, though one word is sufficient to align a reparandum and alteration when the existence of a repair is given, more information is often necessary when the task is not just alignment of repair but also detection of repair. A model that takes into account information sources that identify the existence of repair, such as prosodic cues (Hale et al., 2006; Lickley, 1996), may thus result in improved performance for the simpler unmodeled buffer.

These results also confirm that parsing spontaneous speech with an HHMM can be far superior to a CKY parser, even when the CKY parser is given the advantage of correct POS tags as input. Second, even the baseline HHMM system also improves over the CYK parser in finding edited words, again without the advantage of correct POS tags as input.

In conclusion, the model described here uses a buffer inspired by the phonological loop used in the human auditory system to keep a short memory of recent input. This model, when used to assist in the detection and correction of repair, results in a large increase in accuracy in detection of repair over other most basic parsing systems. This system does not reach the performance levels of lexicalized parsers, nor multi-pass classification systems. Future work will explore ways to apply additional features of these systems or other sources of information to account for the remainder of the performance gap.

References

Alan Baddeley, Susan Gathercole, and Costanza Papagno. 1998. The phonological loop as a language learning device. *Psychological Review*, 105(1):158–173, January.

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

Mark G. Core and Lenhart K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL 99)*.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. ICASSP*, pages 517–520.

John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover, and Robin Stewart. 2006. PCFGs with syntactic and prosodic indicators of speech repairs. In *Proceedings of the 45th Annual Conference of the Association for Computational Linguistics (COLING-ACL)*.

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

Mark Johnson and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, pages 33–39, Barcelona, Spain.

Mark Johnson. 1998a. Finite state approximation of constraint-based grammars using left-corner grammar transforms. In *Proceedings of COLING/ACL*, pages 619–623.

Mark Johnson. 1998b. PCFG models of linguistic tree representation. *Computational Linguistics*, 24:613–632.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430.

Willem J.M. Levelt. 1989. *Speaking: From Intention to Articulation.* MIT Press.

R. J. Lickley. 1996. Juncture cues to disfluency. In *Proceedings of The Fourth International Conference on Spoken Language Processing (ICSLP '96)*, pages 2478–2481.

Tim Miller and William Schuler. 2008. A syntactic time-series model for parsing fluent and disfluent speech. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*.

Kevin P. Murphy and Mark A. Paskin. 2001. Linear time inference in hierarchical HMMs. In *Proc. NIPS*, pages 833–840.

William Schuler, Samir AbdelRahman, Tim Miller, and Lane Schwartz. in press. Broad-coverage incremental parsing using human-like memory constraints. *Computational Linguistics*.

William Schuler. 2009. Parsing with a bounded stack using a model-based right-corner transform. In *Proceedings of the North American Association for Computational Linguistics (NAACL '09)*, Boulder, Colorado.

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

Elizabeth Shriberg. 1996. Disfluencies in Switchboard. In *Proceedings of International Conference on Spoken Language Processing*.