Improvements to a Speech Repair Parser

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Andrew Exley

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Doctor Of Philosophy

April 2016

Improvements to a Speech Repair Parser

by Andrew Exley

ABSTRACT

Parsing is a common task for speech-recognition systems, but many parsers ignore the possibility of speech errors and repairs, which are very common in conversational language. The goal of this thesis is to examine a parsing system that can handle these occurrences and improve its performance by incorporating systems that use linguistic knowledge about speech errors and repairs.

The basis for this thesis is a system for incremental parsing. The thesis shows additions that can be made to that system to allow for detection of speech errors and repairs. That is shown to be an improvement on previous incremental systems.

An extension to the system is introduced which incorporates ideas about human short term memory and its relationship to speech errors. The system is then tested with many different configurations.

Finally, the thesis concludes with a summary and discussion of the various results and lays out possible avenues for future work.

Acknowledgements

I would like to acknowledge many of the people who helped me get to this point.

First, I would like to acknowledge William Schuler for his excellent advice, and his refusal to let me give up even though the process was taking far longer than I would have liked. Further, he was very helpful in developing many of the ideas in this research. In addition, I want to thank Maria Gini for being my secondary adviser and helping me navigate the murky bureaucracy of the University of Minnesota. Also, the other members of my committee, Arindam Banerjee, Sashank Varma and Loren Terveen, asked good questions and were consistently flexible when I needed to meet with them.

I also want to thank members of the bygone Natural Language Processing lab that I shared my time with, including Tim Miller, Stephen Wu, Lane Schwartz, Luan Nguyen, and Dingcheng Li. The lab was a stimulating environment, and a great place to get my research started.

I want to acknowledge my peers at Carleton College, who have consistently encouraged me in this process and been cheerleaders for me, including Jeff Ondich, Dave Musicant, Sherri Goings, Amy Csizmar-Dalal, David Liben-Nowell, Deanna Haunsperger, and all other faculty members who have wished me well.

I also want to thank my students at Carleton, who helped me stay involved in my research and consistently encouraged me in my progress, including Josh Lipstone, Rowan Matney, Sarah Monaghan, and many others who I have encountered in my years

teaching there.

Finally, I want to thank my wife, Megan Smith, for her strong consistent support in the final stages of the process, helping me make difficult decisions and giving me encouragement whenever I needed it.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Spoken language interfaces with computers or other intelligent agents are slowly becoming more common, with many modern smartphones allowing some sort of natural language commands. Being able to talk to one's phone or computer like this is nice, but many of these systems still handle only requests for information that are spoken in clear, error-free sentences. Often this is enough for a simple interface, but in spontaneous human speech, there are a range of possible disruptions (called *disfluencies.*) Spontaneous speech is what occurs in conversations, and it is filled with errors. Any system designed that is designed to understand human conversational speech must be able to deal with these mistakes and still properly parse sentences.

Disfluencies in spontaneous speech include repetitions, filled pauses, parentheticals, word selection errors, non-speech sounds such as coughing or laughter, incomplete fragments, and others. (See table 1.1 for examples.) In addition to these, there are *speech repairs*, in which a speaker realizes their mistake, halts speech and then corrects it by restarting the utterance from a point some number of words earlier, with possible corrections included. These disfluencies and speech repairs are all quite difficult for standard computation models of speech recognition.

Disfluency	**Example**
Filled pauses	yeah *um* I did.
Parentheticals	It was just *you know* a change of location.
Incomplete fragments	Had *t-* place my mother in a nursing home.
Repetitions	So he *he* said yes.
Word selection error	When *I was* my kids were young.

Table 1.1: Examples of speech disfluencies.

This thesis describes the results of experiments for improving an incremental parsing system that can handle speech repairs. The speech repair handling is fully integrated as part of the parser. The current system is better than previous incremental parsing systems, but the results are not yet as good as non-incremental parsers, nor are they as good as systems that separate out the task of repair detection from parsing.

1.1 Motivation

Some speech interface systems ignore speech repairs altogether. It is possible that the systems are still too primitive, or that other parts of such systems are still under development, and have not yet reached a point where a subsystem to handle speech repair is useful. On the other hand, it may also a possibility that ignoring speech repairs altogether is a long-term solution that works well enough. It may be the case that there is a way to set up such systems that would minimize speech repairs, reducing their occurrences to such a low level that it is acceptable if the system cannot handle them. To resolve this question, it is useful to look at the reasons why speech repairs occur, and to take a look at what kinds of systems would exist if interfaces become more complex without attempting to recognize speech repair.

One reason that speech repairs can occur is due to complexity of conversation. If there are many topics that are relevant to the current discussion, or the relations between the topics are particularly complex, a mistake becomes more likely, and thus a repair

becomes more likely. Humans do this quite frequently in complex conversations, and are able to correct their mistakes in a non-ambiguous way and move on. A spoken language system that cannot handle speech repair could minimize that confusion by somehow restricting the domain. This has been done in previous work (Oviatt, 1995), where the user was only allowed to interact with a small subset of the application at a time.

Another reason for speech repairs is distraction of the speaker. Multitasking is common in modern life and often a speaker may become distracted in the middle of an utterance. This occurs often in discourse between humans, and rarely leads to misunderstandings. A user interface that ignores speech repair would require some extraordinary engineering or design to force a user to focus only on that interface in order to avoid disfluencies due to distraction.

A third reason for speech repairs is word selection error. This may happen due to phonological similarity of words, or accidental juxtaposition of words in the sentence. When speakers realize that they have made such a mistake, they will usually perform a speech repair. These errors occur even in the absence of distraction and complexity, and any system designed to reduce these sorts of errors would have to somehow account for basic human mistakes.

In each of these cases, there are suggestions to engineer a system by restricting its domain or somehow forcing a user to maintain attention and focus. In practice, any such "solution" would result in a system that is not viable for conversational speech patterns, and likely would be beyond the scope of many applications. Any system attempting to require the attention and focus of the user would be very difficult to enforce. Engineering the domain that the system uses seems like a reasonable plan, but such an approach would naturally restrict the system away from conversational speech, which would make that system less intuitive.

1.2 Scope

This thesis is concerned with improving parsing of sentences that contain speech disfluencies that occur within a single utterance. Further, the disfluencies considered are ones that contain contiguous fluent speech that then must be deleted to obtain the speaker's intended sentence. These cases include repairs, filled pauses, word selection errors and repetitions. The following sentence contains two examples of disfluencies, in each case the words to be deleted have a line through them:

Because ~~it was you know~~ it was just a ~~change of~~ change of location.

The intended meaning of the sentence is then arrived at as "Because it was just a change of location."

Other types of speech disfluencies are not considered in this work. Nor are non-speech disfluencies, such as laughter or coughing during a speech production.

1.3 Terminology

The terminology used in this paper will follow that of other speech repair work, mainly set out by (Levelt, 1983).

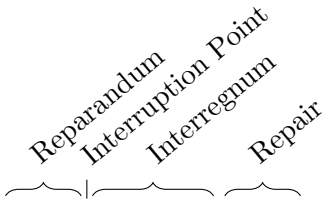

Figure 1.1: A sentence with a speech repair illustrating the terms used in this paper.

1.4 Goal

The goal of the thesis is to present a novel parsing system and extend that system by incorporating information that is intrinsic to speech disfluencies. By specializing parts of the system to deal with speech errors and subsequent repairs, the changes are hoped to increase parsing accuracy, particularly with regards to disfluency detection.

The thesis will attempt many different approaches to working with disfluencies, and then discuss their results.

1.5 Organization

This chapter is the introduction and outline of what is here. Chapter 2 contains a review of previous work in describing, documenting, analyzing and parsing speech disfluencies. Chapter 3 describes the speech parsing system that is the baseline parsing system for the work. Chapter 4 describes changes made to the baseline system that allow it to incorporate speech repairs and their effects on the results. Chapter 5 describes improvements on the speech-repair system. Chapter 6 summarizes the contributions of the entire thesis.

Chapter 2

Related Work and Background

Many others have written work on speech disfluencies, but a large portion of it concerns disfluencies that are not investigated here. As laid out in section 1.2, this proposal only investigates same-turn disfluencies (that is, disfluencies within a single utterance) in spontaneous speech of adult native English speakers, and only those disfluencies in which the intended meaning is arrived at by deleting contiguous segments of speech.

2.1 Early Speech Repair Research

Some of the earliest research in speech repair was published around the turn of the century. In particular, Meringer and Mayer (1895) focused more on phenomena such as spoonerisms, and Freud (1901) contained a section on slips of the tongue.

These early works collected a corpora of speech errors and contained some of the first attempts to explain regularities in the errors. Meringer and Mayer (1895) in particular categorized the errors into types that are still accepted today: *Anticipatory errors* in which a sound from a later part of the utterance is substituted into an earlier part of the same utterance; *preservatory errors* in which a speech sound from an earlier part

of the utterance interferes with later speech articulation; *substitution errors* in which there is no recognizable influence for a speech sound interfering with an utterance; and *exchange errors* in which words or parts of words are exchanged.

Freud's work contains examples of speech errors from many sources in which Freud attempts to explain the error by describing a mental connection between ideas or a speaker's preoccupation that somehow interferes with speech articulation.

Lounsbury (1954), working within the idea of a hierarchical speech generation and understanding, examined the role of hesitations in production of speech. He proposed five hypotheses explaining the relationship between hesitations, structure and "units of encoding" in speech production, the first two of which concern hesitations, and are discussed here. A "unit of encoding" here is a unit within the hierarchical encoding process, and it is potentially some intermediate representation of syntax or semantics that may or may not align with the boundaries of words or phonemes.

The first hypothesis is that hesitations occur more frequently between words (or units) that have high statistical uncertainty. In essence, the idea is that the less likely that two words, morphemes or phonemes are to following one another, the more likely the hesitation.

The second hypothesis is that hesitation frequency and points of high statistical uncertainty correspond to the beginning of units of encoding. This hypothesis claims that transitions to new units of encoding cause some sort of short-term disfunction in the speech production apparatus.

Maclay and Osgood (1959) examined several disfluency types in an annotated corpus of spontaneous speech. They classified disfluencies as *repeats*, *filled pauses*, *unfilled pauses* and *false starts*, and examined the distribution of each of these types in their corpus.

Among their findings was that both filled and unfilled pauses are less likely to occur

in front of function words (determiners, prepositions, etc.,) than in front of lexical words (nouns, verbs, etc.)

These early efforts at understanding the psychological causes of speech errors are important for computational models. Building a model that relies on a generative approach requires an understanding of what causes speech errors so that they may even be considered as a possibility to generate.

2.2 Recent Study of Speech Repair

Levelt (1983) is the most widely-used reference for modern speech repair research. In his work, he documents nearly a thousand separate instances of speech repairs in spontaneous speech. The speech was generated by asking speakers to describe certain visual patterns involving lines and colored dots. The patterns were to be described in such a way that a listener could reproduce the pattern from the audio description. The result was that often speakers would make mistakes as they were describing scenes, or realize during their utterance that their description was perhaps not accurate enough. Any time that this occured, a speech disfluency was generated and Levelt was able to capture disfluency information and statistically analyze the results to evaluate claims about speech repair and processing

Levelt's paper declared two rules regarding speech repair. The first is the *Main Interruption Rule*, which states that speakers will "stop the flow of speech immediately upon detecting the occasion of repair." That is, as soon as a speaker detects that a repair is necessary, that speaker halts their utterance and then begins to consider the correct way to repair the utterance. Speakers will not necessarily complete the constituent that they are currently in the middle of uttering.

Levelt's other important rule is the *Well-formedness rule*. This rule describes that

repairs and reparanda have a relationship similar to two constituents joined by a coordination. To quote Levelt:

> A repair $\langle \alpha\ \gamma \rangle$ is well-formed if and only if there is a string β such that the string $\langle \alpha\ \beta$ and* $\gamma \rangle$ is well-formed, where β is a completion of the constituent directly dominating the last element of α. (**and* to be deleted if γ's first element is itself a sentence connective.)

Levelt's statistical analysis showed that nearly all of his collected examples fit this rule.

2.3 Interregnum Analysis

Some studies have gone into analyzing the types of utterances that occur when a repair is underway. Often, once a speaker has realized an error was made and a repair is required, the speaker will continue to speak, even though the repair itself has not yet begun. As described in the previous chapter, this is known as the interregnum.

2.3.1 Filled Pauses

Certain filler words, such as *um*, *uh*, *er*, *ah*, and so on, were, as a group, labelled as interjections by early linguists, and not considered to be very different from one another. However, more recent research, especially as compiled by Clark and Fox Tree (2002) shows that these different fillers can have different meanings, for example *um* and *uh* are often used to signal different lengths of pauses, perhaps due to a speaker's estimate of how complicated a repair will be.

Arnold et al. (2003) also examines the semantic content of filled pauses. Their experiments involved scenes of objects, some with initially ambiguous names, and commands to participants regarding those objects. Some of the commands contained filled pauses, and others did not. By monitoring the subjects' eye movements the experiment was

able to examine the effect of these filled pauses on visual fixation. The results showed that in cases where filled pauses preceded an object reference, subjects visually fixated on the newer object. Thus, the study suggests that filled pauses signal to a listener that upcoming information is new, and perhaps that the listener should decrease the expectation of coreference.

2.3.2 Repeated Words

Another common occurrence in disfluencies is repetition of single words. In the Switchboard corpus of spontaneous speech, 8.6% of the sentences in the training set contain repeated words (Godfrey et al., 1992). When a single word is repeated multiple times, the disfluency can be treated as multiple repairs, or that the word repetition is simply a type of filled pause, with copies of the repeated word making up part of the interregnum.

Clark and Wasow (1998) examines closely repeated words in disfluencies. They show that speakers are more likely to repeat initial determiners of complex NP constituents than of simple ones. Repetition of pronouns is much more likely when the pronoun occurs at the left edge of a clause, and again, more often when the clause is more complicated. This observation may be of use in a computational model which could use a repeated word as likely evidence that an upcoming constituent is more likely to be complex.

2.4 Memory in Language

Psychological models of memory (Atkinson and Shiffrin, 1968) propose a short-term memory that retains around 7 units of unique information. Later models (Baddeley and Hitch, 1974) attempted to more accurately describe how this short-term memory worked, by breaking it down into detailed subsystems. One of the described subsystems is the *phonological loop*, which is responsible for dealing with phonological information.

Support for the existence of this particular subsystem was given in (Madigan and McCabe, 1971), in which subjects show a difference in recall when presented with lists in audio form as opposed to visual form. Botvinick (2007) discusses the hierarchical nature of events and sequential behavior and proposes that this corresponds to a neurological model that is itself hierarchic. Botvinick (2007) does this by examining a previously proposed computational model, (Botvinick and Plaut, 2004), based on ideas developed in earlier memory models (Fuster, 1990, 1997, 2001). The model is implemented as a recurrent neural network. While the model itself did not contain hierarchical structure, it seemed to form hierarchies after being trained on certain tasks, where particular nodes in the neural network would become responsible for special roles in representing task context.

There are other psychological models of memory (Ericsson and Kintsch, 1995, Cowan, 1999), and the debate about the most appropriate model is far from over, but all of them contain a specialized phonological memory system. Therefore, in our system that attempts to use human-like recognition and parsing techniques, it seems reasonable to attempt to model some sort of interaction between memory and the phonological recognition system with a hierarchic structure.

2.5 Computational Models of Repairs

Using statistical information from earlier studies, multiple systems have been constructed that use speech data to attempt to detect speech repairs.

Johnson and Charniak (2004) propose a model that uses a so-called "noisy channel" approach to detect speech repairs. The system is based on a Tree-Adjoining Grammar (TAG) parser and uses the fact that repairs are often very similar to the reparanda to detect speech repairs.

Another such model by Hale et al. (2006) exploits combined syntactic and prosodic

indicators in speech to improve speech repair detection. The model shows that pauses in speech production are very clear indicators that speech repair is imminent, and also uses syntactic information (on the basis of Levelt's well-formedness rule) that repairs are similar in structure to their reparanda.

Miller (2009) uses an explicit model of the phonological loop to retain information about previously used words and increase probabilities of those words and structures re-occurring when speech repairs are encountered. This buffer model introduces two new random variables to his system, one that models interruption points, and one that explicitly models a buffer of up to 4 previously spoken words.

Chapter 3

Baseline System

This chapter describes a parser that is based on a domain-general hierarchic sequence prediction framework. The system will be the basis upon which later chapters will build to allow for disfluencies and improve upon disfluency detection.

The system treats each time step in a sequence of linguistic events as an active superordinate event lacking a possibly complex awaited subordinate event yet to come. It is based on the idea that humans, when perceiving sequences (especially language) will make assumptions about goal states and predictions regarding upcoming items in a sequence.

These active and awaited events delimit *connected components* of nested event structures. Section 3.1 describes the connected component systems and how we use this idea to form a general sequence prediction of superordinate and subordinate events given a series of observations. Section 3.2 describes how this system works when it is applied to the specific problem of natural language parsing, and shows some simple examples.

Section 3.3 introduces probability to the connected components system. Sections 3.4 through 3.6 introduce Probabilistic Context-Free Grammars (PCFGs) and describe a way to convert a grammar described as a PCFG into a set of probabilistic rules for

a connected components system. Section 3.7 describes the PCFG grammar training algorithm used by many state-of-the-art parsers, which we use in training before transforming to a hierarchic sequential prediction model.

Finally, section 3.8 describes the results of this system on standard parsing tasks and compares it with other parsing systems.

3.1 Domain-General Sequence Prediction With Connected Components

The model represents time in discrete steps t, corresponding to discrete observations of simple events x_t. At each time step, the model maintains several hypotheses q_t which are probabilistically weighted and considered in parallel. Each hypothesis defines a hierarchy of disjoint connected component states q_t^d in a set of nested state sequences, orded by depth d from superordinate ($d = 1$) to subordinate ($d > 1$). Each connected component state $q_t^d = a_t^d/b_t^d$ defines a maximal connected component in a graph of relations among events up to the current time step, consisting of an *active* goal event a_t^d lacking an *awaited* event b_t^d yet to come.

Descriptions of the sequential prediction model are defined in terms of a set of structured event relations of the form $a \rightarrow a'\ b'$ (meaning that event a is equivalent to an initial child event a' followed by event b',) or of the form $a \rightarrow x$ (meaning simple event a is associated directly with observation x). In addition to this set of structured event relations, this model also assumed an ability to predict (possibly deeply nested) initial sub-events a' of larger events b, denoted $b \xrightarrow{+} a'$.

A simple nondeterministic process for incrementally predicting event structures using a sequence of connected component states can be defined as a deductive system,

given an input sequence consisting of an initial connected component state $\top/\top$, followed by a sequence of observed events $x_1, \ldots, x_n$, processed in time order. As each x_t is encountered, it either connects to the existing components, or it introduces a new disjoint component using productions that treat each word as the first observation of a terminated connected component state, or as neither, or as both.

The model assumes that all operations are conditioned on the current connected component and the most recent prior connected component at the beginning of the current time step.

First, if an observation x_t can attach as the awaited event b of the most recent (most superordinate) connected component a/b from a to b, it is hypothesized to do so, turning the incomplete a/b into the complete a (via Production F–)[1] . If the observation x_t can serve as the lower descendant of the awaited event b, it is hypothesized to form the first event a' in a newly initiated complete connected component (via Production F+) .

$$\frac{a/b \quad x_t}{a} \quad b \rightarrow x_t \tag{F–}$$

$$\frac{a/b \quad x_t}{a/b \quad a'} \quad b \xrightarrow{+} a' \ldots; \quad a' \rightarrow x_t \tag{F+}$$

Then, if either of these complete events (a or a', matched to a'' below) can immediately attach as an initial child of the awaited event of the most recent connected component a/b, it is hypothesized to merge and extend this connected component, with x_t as the last observation of the completed connected component (L+); or if it can serve as a lower descendant of this awaited event, it is hypothesized to remain disjoint and form

[1] The productions here use the notation of a deductive system, where $\frac{P}{Q}R$, means premise P entails conclusion Q according to rule R.

its own connected component (L–):

$$\frac{a/b \quad a''}{a/b''} \; b \rightarrow a'' \; b'' \tag{L+}$$

$$\frac{a/b \quad a''}{a/b \quad a'/b''} \; b \stackrel{+}{\rightarrow} a''...; \quad a' \rightarrow a'' \; b'' \tag{L–}$$

These initiation (F) and termination (L) productions are similar to the push and pop operations respectively of a nondeterministic pushdown automaton or the shift and reduce operations of a shift-reduce parser, except that the model has a memory-based processing constraint that it can use no more than one initiation (F) and one termination (L) production per time step.

3.2 Connected Components Applied to Parsing

When applied to parsing, the events are part-of-speech tags, such as Noun (N), Determiner (D), Noun Phrase (NP), Verb (V), Verb Phrase (VP), Sentence (S), etc. Thus, given a subset of a simple grammar, there are events which generate observations (i.e. terminal productions)

- D $\rightarrow$ *the*
- N $\rightarrow$ *cat*
- N $\rightarrow$ *mouse*
- V $\rightarrow$ *ate*

and events that generate a sequence of subevents (i.e. nonterminal productions.)

- NP $\rightarrow$ D N

- VP $\rightarrow$ V NP
- S $\rightarrow$ NP VP

Using this simple grammar, and the connected component system, here is an example of the system parsing the sentence *the cat ate the mouse.* Each step will show an initiation and termination production. The sequence begins with the initial connected component state $\top/\top$, and each word of the sentence is an observation. $\top \rightarrow$ S $\top$ is added as an implicit nonterminal production.

1. Observation x_1 *the*: D could be an initial sub-event of $\top$ ($\top \xrightarrow{+}$ D ...) and the event D $\rightarrow$ *the* allows for an F+ production. Then, the model attempts to apply an L rule. There is no rule that fits an L+ production, but the same $\top \xrightarrow{+}$ D ... and the rule NP $\rightarrow$ D N is used to generate the disjoint component NP/N using an L– production.

$$\dfrac{\dfrac{\top/\top \quad \text{the}}{\top/\top \text{ D}}\ \text{F+}}{\top/\top \text{ NP/N}}\ \text{L–}$$

2. Observation x_2 *cat*: Production F– uses the rule N $\rightarrow$ *cat* to turn NP/N into NP. Then production L– uses $\top \xrightarrow{+}$ NP ... and S $\rightarrow$ NP VP to generate S/VP.

$$\dfrac{\dfrac{\top/\top \text{ NP/N} \quad \text{cat}}{\top/\top \text{ NP}}\ \text{F–}}{\top/\top \text{ S/VP}}\ \text{L–}$$

3. Observation x_3 *ate*: Production F+ uses the rule V $\rightarrow$ *ate* and VP $\xrightarrow{+}$ V ... to generate a new disjoint V. Then, production L+ combines S/VP and V using VP $\rightarrow$ V NP to generate S/NP.

$$\dfrac{\dfrac{\text{ate}}{\top/\top \text{ S/VP V}}\ \text{F+}}{\top/\top \text{ S/NP}}\ \text{L+}$$

4. Observation x_4 *the*: Production F+ uses the rule D $\rightarrow$ *the* and NP $\rightarrow$ N ... to generate a new disjoint D. Then, production L+ combines S/NP and D using NP $\rightarrow$ D NP to generate S/N.

$$\dfrac{\dfrac{\text{the}}{\top/\top\ \text{S/NP D}}\ \text{F+}}{\top/\top\ \text{S/N}}\ \text{L+}$$

5. Observation x_5 *mouse*: Production F– uses the rule N $\rightarrow$ *mouse* to generate S from S/N. Finally, production L+ uses $\top \rightarrow$ S $\top$ to generate $\top/\top$.

$$\dfrac{\dfrac{\text{mouse}}{\top/\top\ \text{S}}\ \text{F–}}{\top/\top}\ \text{L+}$$

If there is no bound on the number of disjoint connected component states that can be hypothesized in a hierarchy, this system is able to generate all and only those syntactic structures allowed by its binary context-free grammar rules. This can be shown by observing that (i) every binary syntactic tree over t observations must also contain t branches from a parent event to a pair of children (including a branch connecting this tree to a generally right-branching discourse,) and (ii) for each observation x_t in a complete tree there is a unique largest event which is co-final with x_t. Inspection of the F and L operations shows that F– and F+ isolate this largest co-final event (as a in the consequent of F– or a' in the consequent of F+), and L– and L+ connect this event (as a'') to a parent or sibling in some tree (to b and b'' in L+ and to a'' and b'' in L–). Since each such connection is unique, and since there are only t such connections in any complete tree, the system must be able to predict any complete tree for any sequence of t observations.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\top/\top \quad \text{the}}{\top/\top,\ \text{D}}\ \text{F+}}{\top/\top,\ \text{NP/N}}\ \text{L–} \quad \text{cat}}{\top/\top,\ \text{NP}}\ \text{F–}}{\top/\top,\ \text{S/VP}}\ \text{L–} \quad \text{ate}}{\top/\top,\ \text{S/VP, V}}\ \text{F+}}{\top/\top,\ \text{S/NP}}\ \text{L+} \quad \text{the}}{\top/\top,\ \text{S/NP, D}}\ \text{F+}}{\top/\top,\ \text{S/N}}\ \text{L+} \quad \text{mouse}}{\top/\top,\ \text{S}}\ \text{F–}}{\top/\top}\ \text{L+}$$

Figure 3.1: The complete derivation of the sentence *the cat ate the mouse* using F–, F+, L–, and L+ productions.

3.3 Probabilistic Parsing

This process can then be extended to calculate probabilities for event structures by introducing distributions ϕ and λ over the binary F and L decisions defined above, with constraints taken from the following productions.

F decisions (about whether to introduce a new subordinate sequence) are constrained such that:

$$\mathsf{P}_\phi(\text{`}-\text{'} \mid b\ x) \neq 0 \quad \text{only if} \quad b \to x \tag{3.1a}$$

$$\mathsf{P}_\phi(\text{`}+\text{'} \mid b\ x) \neq 0 \quad \text{only if} \quad b \xrightarrow{+} a' \ldots \ \text{ and } \ a' \to x \ \text{ for some } a' \tag{3.1b}$$

L decisions (about whether to terminate a subordinate sequence) are constrained such that:

$$\mathsf{P}_\lambda(\text{`}+\text{'} \mid b\ a'') \neq 0 \quad \text{only if} \quad b \to a''\ b'' \ \text{ for some } b'' \tag{3.2a}$$

$$\mathsf{P}_\lambda(\text{`}-\text{'} \mid b\ a'') \neq 0 \quad \text{only if} \quad b \xrightarrow{+} a' \ \ldots \ \text{ and } \ a' \to a''\ b'' \ \text{ for some } \ a', b'' \tag{3.2b}$$

Constraints for distributions α and β over the active and awaited events a and b in

hypothesized connected component states are also derived from the F and L productions:

$$\mathsf{P}_{\alpha}(a' \mid b\ a'') \neq 0 \quad \text{only if} \quad b \xrightarrow{+} a' \ldots \text{ and } a' \rightarrow a''\ b'' \text{ for some } b'' \tag{3.3}$$

$$\mathsf{P}_{\beta}(b'' \mid a'\ a'') \neq 0 \quad \text{only if} \quad a' \rightarrow a''\ b'' \tag{3.4}$$

Since F productions take observations x as input and produce complete events a as output, and L productions take complete events a as input and produce connected component states a/b as output, the process may only iterate by applying exactly one F production and one L production at each time step.

As F and L productions each have two ('+' and '–') options, a complete hierarchic sequential prediction model σ can be defined using only four cases: one for each combination of F and L productions.

These cases are represented as addends in the definition below. Each addend is a product of factors for:

1. hypothesizing a combination of an initiation and a termination of a subordinate state sequence (using ϕ and λ probabilities),

2. hypothesizing an active and awaited event for the most subordinate connected component state in the resulting hierarchy (using α and β probabilities), and

3. deterministically carrying forward unused or unmodified states from the previous time step.

All models depend on the depth d' of the most subordinate connected component state at the previous time step, and (in the case of the β model) on whether the first parameter is an active or awaited event ('A' or 'B', respectively). In this definition, D is an arbitrary bound on the size of the state hierarchy (set to 4 in these implementations), and $[\![...]\!]$ is a deterministic indicator function, evaluating to 1 if '...' is true, and 0

otherwise, used to represent a deterministic distribution:

$$
\begin{aligned}
&\mathsf{P}_\sigma(q_t^{1..D} \mid q_{t-1}^{1..D}\ x_{t-1}) \\
&\quad \stackrel{\text{def}}{=} \mathsf{P}_{\phi_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot [\![a_t^{d'-1} = a_{t-1}^{d'-1}]\!] \cdot \mathsf{P}_{\beta_{\mathrm{B},d'-1}}(b_t^{d'-1} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \\
&\qquad \cdot [\![q_t^{1..d'-2} = q_{t-1}^{1..d'-2}]\!] \cdot [\![q_t^{d'..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot \mathsf{P}_{\alpha_{d'}}(a_t^{d'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot \mathsf{P}_{\beta_{\mathrm{A},d'}}(b_t^{d'} \mid a_t^{d'}\ a_{t-1}^{d'+1}) \\
&\qquad \cdot [\![q_t^{1..d'-1} = q_{t-1}^{1..d'-1}]\!] \cdot [\![q_t^{d'+1..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot [\![a_t^{d'} = a_{t-1}^{d'}]\!] \cdot \mathsf{P}_{\beta_{\mathrm{B},d'}}(b_t^{d'} \mid b_{t-1}^{d'}\ x_{t-1}) \\
&\qquad \cdot [\![q_t^{1..d'-1} = q_{t-1}^{1..d'-1}]\!] \cdot [\![q_t^{d'+1..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\alpha_{d'+1}}(a_t^{d'+1} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\beta_{\mathrm{A},d'+1}}(b_t^{d'+1} \mid a_t^{d'+1}\ x_{t-1}) \\
&\qquad \cdot [\![q_t^{1..d'} = q_{t-1}^{1..d'}]\!] \cdot [\![q_t^{d'+2..D} = \text{`}-\text{'}]\!] \qquad (3.5)
\end{aligned}
$$

Note that the active event of a superordinate state is deterministically carried forward whenever x_t is the last event in a subordinate state sequence (when λ is positive).

This is because the active event of a superordinate state does not change when a subordinate state is completed.

These prediction probabilities σ are then combined with observation probabilities ξ to define a most likely sequence of connected component hierarchies $\hat{q}_{1..T}^{1..D}$:

$$
\hat{q}_{1..T}^{1..D} \stackrel{\text{def}}{=} \operatorname*{argmax}_{q_{1..T}^{1..D}} \prod_{t=1}^{T} \mathsf{P}_\sigma(q_t^{1..D} \mid q_{t-1}^{1..D}\ x_{t-1}) \cdot \mathsf{P}_\xi(x_t \mid b_t^{d'}) \qquad (3.6)
$$

This model predicts a syntactic tree structure while restricting access to superordinate states as a memory-based recency constraint. Since the model is implemented as a sum of products, it is essentially equivalent to a localist representation in a recurrent neural network, with a hidden context unit for every combination of disjoint connected components, represented in an explicit state hierarchy. However, in order to maintain a close connection with linguistic notions of syntactic structure, the model is not trained using unsupervised learning techniques traditionally applied to connectionist models.

3.3.1 Implications of an Arbitrary Depth Bound

As mentioned, there is an arbitrary bound D, which is set to 4 in these implementations. The previous section pointed out that without a bound, the system can generate all rules that can be created from a binary CFG. But if an arbitrary bound is imposed, then certain trees will not be able to be generated by the system.

It turns out that disjoint connected components are created only by certain structures in the CFG, which correspond to center embedding. An example of this is the classic set of clauses: If *the cart the horse pulled broke*, is legal and *the horse the man bought* is also a valid noun phrase, then what about the sentence, *the cart the horse the man bought pulled broke.* Figures 3.2 and 3.3 show the trees for these clauses.

Most human listeners declare the third sentence *the cart the horse the man bought pulled broke* to be ungrammatical. A further examination of the Wall Street Journal corpus (**?**) showed that a depth limit of 4 was sufficient to cover nearly all the syntax trees in the corpus. Given this evidence it seemed that not only was a depth bound not going to hurt the model, it might potentially help by reducing the possibility of the model to generate some sequences that humans would consider ungrammatical.

3.4 Probabilistic Context Free Grammars

The constraints described in the previous section can be satisfied in a variety of ways. The system used here is directly defined over a Probabilistic Context Free Grammar (PCFG), trained using latent variable induction (Petrov et al., 2006). PCFGs are widely used in parsing because they provide a simple branching stochastic process (Collins, 1997), because well-studied algorithms exist for inferring or refining PCFGs from data (Petrov et al., 2006), and because PCFGs have been shown to be useful as a basis for information-theoretic accounts of garden path effects and reading time delays (Jurafsky,

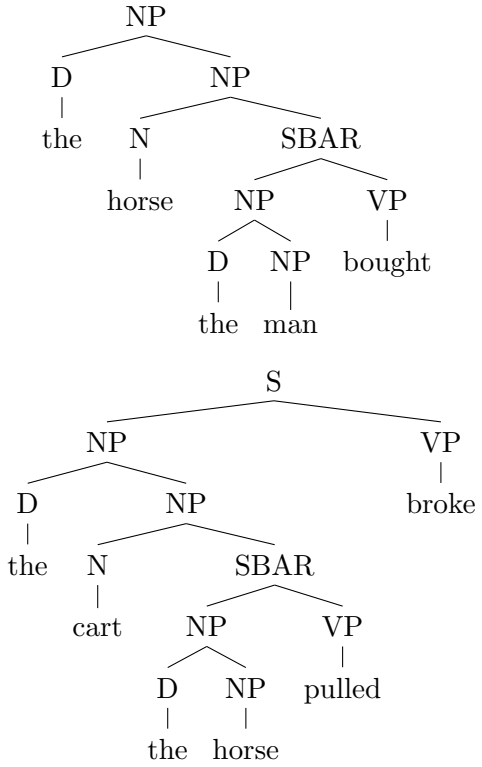

Figure 3.2: Examples of clauses with center-embedding.

1996, Hale, 2001, 2003, 2006, Levy and Jaeger, 2007).

3.4.1 Side- and Depth-Specific Rules

The general hierarchic sequence model described in the previous section can be defined for a given PCFG γ by first deriving side- and depth-specific rule probabilities and expected counts of initial descendant sub-events from rule probabilities in Chomsky Normal Form (CNF).

The model imposes hard constraints on the number of disjoint syntactically connected components allowed in each hypothesis. This has an effect of bounding the number of center embeddings allowed in any partial syntactic tree (in particular, initial

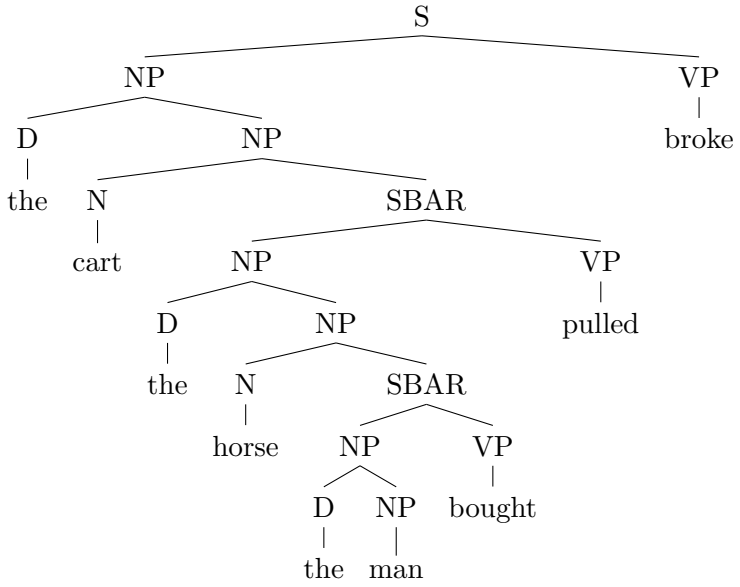

Figure 3.3: A sentence with center embedding that is difficult for humans to parse.

children of final children in any CNF derivation). This is done by first computing a side- and depth-specific PCFG 'fit' model $\delta^{(i)}_{s,d}$, defining the probability that a subtree below an event of category b, occurring on its parent's initial or final side $s \in \{A, B\}$, will fit within a bounded depth d of disjoint connected components. This fit model is computed according to the following recursive definition, where i is the recursive iteration:

$$\mathsf{P}_{\delta^{(0)}_{s,d}}(1 \mid b) \stackrel{\text{def}}{=} 0 \tag{3.7a}$$

$$\mathsf{P}_{\delta^{(i)}_{A,d}}(1 \mid b) \stackrel{\text{def}}{=} \sum_{x} \mathsf{P}_{\gamma}(b \to x) + \sum_{a',b'} \mathsf{P}_{\gamma}(b \to a'\ b') \cdot \mathsf{P}_{\delta^{(i-1)}_{A,d}}(1 \mid a') \cdot \mathsf{P}_{\delta^{(i-1)}_{B,d}}(1 \mid b') \tag{3.7b}$$

$$\mathsf{P}_{\delta^{(i)}_{B,d}}(1 \mid b) \stackrel{\text{def}}{=} \sum_{x} \mathsf{P}_{\gamma}(b \to x) + \sum_{a',b'} \mathsf{P}_{\gamma}(b \to a'\ b') \cdot \mathsf{P}_{\delta^{(i-1)}_{A,d+1}}(1 \mid a') \cdot \mathsf{P}_{\delta^{(i-1)}_{B,d}}(1 \mid b') \tag{3.7c}$$

Note that the only difference between the initial-event ($\delta^{(i)}_{A,d}$) and final-event ($\delta^{(i)}_{B,d}$) cases above is simply that the depth is incremented for initial children of final children. These initial-final zig-zags form the breaks between connected components in the hierarchy. In practice the recursive product is estimated to some constant I using value iteration

(Bellman, 1957).

Now a side- and depth-specific PCFG model $\gamma_{s,d}$ can be defined by renormalizing over the probability mass isolated in $\delta^{(I)}_{s,d}$:

$$\mathsf{P}_{\gamma_{A,d}}(b \to a'\ b') \stackrel{\text{def}}{=} \frac{\mathsf{P}_{\gamma}(b \to a'\ b') \cdot \mathsf{P}_{\delta^{(I)}_{A,d}}(1 \mid a') \cdot \mathsf{P}_{\delta^{(I)}_{B,d}}(1 \mid b')}{\mathsf{P}_{\delta^{(I)}_{A,d}}(1 \mid b)} \tag{3.8a}$$

$$\mathsf{P}_{\gamma_{B,d}}(b \to a'\ b') \stackrel{\text{def}}{=} \frac{\mathsf{P}_{\gamma}(b \to a'\ b') \cdot \mathsf{P}_{\delta^{(I)}_{A,d+1}}(1 \mid a') \cdot \mathsf{P}_{\delta^{(I)}_{B,d}}(1 \mid b')}{\mathsf{P}_{\delta^{(I)}_{B,d}}(1 \mid b)} \tag{3.8b}$$

This renormalizing over $\delta^{(I)}_{s,d}$ ensures no probability mass is lost when the depth of the model is bounded. Again, the only difference between the initial- and final-event cases is that the depth is incremented for initial children of final children.

3.4.2 Initial Descendant Sub-Event Expected Counts

The model also needs initial descendant sub-event expected counts, which are based on the expected number of times an event of category a'' occurs at the beginning of an event of category b after any number of expansions. This is also estimated recursively with j as the recursive iteration:

$$\mathsf{E}_{\gamma^*_d}(b \xrightarrow{1} a'\ ...) \stackrel{\text{def}}{=} \sum_{b'} \mathsf{P}_{\gamma_{B,d}}(b \to a'\ b') \tag{3.9a}$$

$$\mathsf{E}_{\gamma^*_d}(b \xrightarrow{j} a''\ ...) \stackrel{\text{def}}{=} \sum_{a',b''} \mathsf{E}_{\gamma^*_d}(b \xrightarrow{j-1} a'\ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \to a''\ b'') \tag{3.9b}$$

$$\mathsf{E}_{\gamma^*_d}(b \xrightarrow{+} a''\ ...) \stackrel{\text{def}}{=} \sum_{j=1}^{\infty} \mathsf{E}_{\gamma^*_d}(b \xrightarrow{j} a''\ ...) \tag{3.9c}$$

$$\mathsf{E}_{\gamma^*_d}(b \xrightarrow{*} a''\ ...) \stackrel{\text{def}}{=} [\![b{=}a'']\!] + \mathsf{E}_{\gamma^*_d}(b \xrightarrow{+} a''\ ...) \tag{3.9d}$$

Here again, the recursive products and infinite sum are estimated to some constant J using value iteration (Bellman, 1957).

3.5 Transition Operations for Language Comprehension

The model probabilities are then just straightforward probabilistic implementations of the constraints specified in Equations 3.1a–3.4 expressed in terms of side- and depth-specific rule probabilities and expected counts of initial descendant sub-events from a PCFG γ as described in the previous subsection:

1. The initiation model ϕ probabilities are calculated from the expected counts of an event of syntactic category a' occurring as an initial descendant sub-event of a larger event of category b multiplied by the probability of that category generating an observation x:

$$\mathsf{P}_{\phi_d}(\text{`}-\text{'} \mid b\ a') \stackrel{\text{def}}{=} \frac{[\![b = a']\!] \cdot \sum_x \mathsf{P}_\gamma(a' \rightarrow x)}{\mathsf{E}_{\gamma^*_d}(b \stackrel{*}{\rightarrow} a'\ ...) \cdot \sum_x \mathsf{P}_\gamma(a' \rightarrow x)} \tag{3.10a}$$

$$\mathsf{P}_{\phi_d}(\text{`}+\text{'} \mid b\ a') \stackrel{\text{def}}{=} \frac{\mathsf{E}_{\gamma^*_d}(b \stackrel{+}{\rightarrow} a'\ ...) \cdot \sum_x \mathsf{P}_\gamma(a' \rightarrow x)}{\mathsf{E}_{\gamma^*_d}(b \stackrel{*}{\rightarrow} a'\ ...) \cdot \sum_x \mathsf{P}_\gamma(a' \rightarrow x)} \tag{3.10b}$$

2. The termination model λ probabilities are calculated from the expected counts of an event of syntactic category a' occurring as an initial descendant sub-event of a larger event of category b multiplied by the probability of that event having an initial child of category a'':

$$\mathsf{P}_{\lambda_d}(\text{`}+\text{'} \mid b\, a'') \stackrel{\text{def}}{=} \frac{\sum_{a',b''} [\![b = a']\!] \cdot \mathsf{P}_{\gamma_{B,d}}(a' \rightarrow a''\ b'')}{\sum_{a',b''} \mathsf{E}_{\gamma^*_d}(b \stackrel{*}{\rightarrow} a'\ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''\ b'')} \tag{3.11a}$$

$$\mathsf{P}_{\lambda_d}(\text{`}-\text{'} \mid b\, a'') \stackrel{\text{def}}{=} \frac{\sum_{a',b''} \mathsf{E}_{\gamma^*_d}(b \stackrel{+}{\rightarrow} a'\ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''\ b'')}{\sum_{a',b''} \mathsf{E}_{\gamma^*_d}(b \stackrel{*}{\rightarrow} a'\ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''\ b'')} \tag{3.11b}$$

3. The active event model α probabilities are calculated from the expected counts of an event of syntactic category a' occurring as an initial descendant sub-event of a larger event of category b multiplied by the probability of that event having an

initial child of category a'':

$$\mathsf{P}_{\alpha_d}(a' \mid b\, a'') \stackrel{\text{def}}{=} \frac{\sum_{b''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''\ b'')}{\sum_{a',b''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ ...) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''\ b'')} \tag{3.12}$$

4. The awaited event model β probabilities are simply the probability that an event of category a has a final child of category b' given that it has an initial child of category a':

$$\mathsf{P}_{\beta_{s,d}}(b' \mid a\, a') \stackrel{\text{def}}{=} \frac{\mathsf{P}_{\gamma_{s,d}}(a \rightarrow a'\ b')}{\sum_{b'} \mathsf{P}_{\gamma_{s,d}}(a \rightarrow a'\ b')} \tag{3.13}$$

3.6 Transition Model for Language Comprehension

These individual model probabilities are combined into a single connected component transition probability σ, as described in Equation 3.5 of the previous section. These connected component transition probabilities σ are then combined with preterminal and terminal probabilities π and ξ, described below.

In the previous section, observations were generated directly from awaited events. In practice, it is more efficient to make the assumption that certain syntactic categories are preterminals, which generate a single lexical observation as a subevent. These preterminals, then, are the predictions of the parsing system (simple events) which in language processing are subsequently grounded by the observation of a lexical item. Such an assumption is made primarily for efficiency since only a subset of syntactic categories must then be considered for prediction, which reduces the complexity of the ϕ, α, and β models that depend on simple events. Formally, the preterminal probabilities π define the normalized probabilities of generating a simple event p as an initial subevent of the syntactic category b:

$$\mathsf{P}_{\pi_d}(p \mid b) \stackrel{\text{def}}{=} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{*} p \ ...) \cdot \sum_x \mathsf{P}_{\gamma}(p \rightarrow x) \tag{3.14}$$

Terminal probabilities ξ are then defined as the normalized probabilities of generating an observation x from a preterminal p:

$$\mathsf{P}_\xi(x \mid p) \stackrel{\text{def}}{=} \frac{\mathsf{P}_\gamma(p \rightarrow x)}{\sum_x \mathsf{P}_\gamma(p \rightarrow x)} \tag{3.15}$$

These transition probabilities σ, preterminal probabilities π, and terminal probabilities ξ are then combined to define a most likely sequence $\hat{q}_{1..T}^{1..D}$:

$$\hat{q}_{1..T}^{1..D} \stackrel{\text{def}}{=} \underset{q_{1..T}^{1..D}}{\operatorname{argmax}} \prod_{t=1}^{T} \mathsf{P}_\sigma(q_t^{1..D} \mid q_{t-1}^{1..D}\, p_{t-1}) \cdot \mathsf{P}_{\pi_{d'}}(p_t \mid b_t^d) \cdot \mathsf{P}_\xi(x_t \mid p_t); \quad d \stackrel{\text{def}}{=} \max\{d' \mid q_{t-1}^{d'} \neq \text{`–'}\} \tag{3.16}$$

3.7 Training

The system uses the split-merge-smooth algorithm from Petrov et al. (2006) to extract a latent variable PCFG from the Penn Treebank corpus. The corpus uses a set of category labels for characterization of syntactic constituents. The split-merge-smooth algorithm attempts to find subcategorizations (splits) of category labels which conform to distinct distributions in the set of training data. For example, the class of present tense ditransitive verbs (such as *gives*) may be discovered to have a different distribution than the class of present tense transitive verbs (such as *owns*), though both are typically labeled 'VBZ' (present tense verb) in Treebank's labeling scheme. Another iteration of the algorithm may then find that certain categories appear more often as first arguments of ditransitive verbs (now that they have been distinguished) than as arguments of transitive verbs. Assigning each such distribution a unique subcategory label helps encode mild contextual information into each label. To avoid overfitting training data, splits which are not sufficiently statistically informative are then merged back into a larger category. The PCFG relations used in the previous section are then calculated over these refined grammar categories.

3.8 Experimental Results

The accuracy of this parser was compared to that of the Petrov and Klein (2007) and Roark (2001) parsers using varying numbers of competing hypotheses (referred to as *beam width*). The Petrov and Klein (2007) parser is a chart parser based on the same latent variable PCFG Petrov et al. (2006) used to define the hierarchic sequence model for the system described here. The Roark (2001) parser is an incremental parser widely used in cognitive modeling evaluations.

The evaluated hierarchic sequence model restricts the number of disjoint connected components in any hypothesis to at most four. This limit was empirically determined to be sufficient to achieve greater than 99.9% coverage on the Wall Street Journal corpus (Schuler et al., 2010).

All parsers were trained on Sections 02-21 of the Wall Street Journal portion of the Penn Treebank (Marcus et al., 1993) and tested on Section 23. With the exception of the (Roark, 2001) parser, all parsers used 5 iterations of the Petrov et al. (2006) split-merge-smooth algorithm. This is the recommended number of iterations of the algorithm to get high accuracy while avoiding overfitting (Petrov and Klein, 2007).

This corpus consists of sentences from many sections of the Wall Street Journal, including opinion and editorial pieces. The sentences range from short and simple to quite long and complex. Some excerpts from file 2377 (which is in section 23 of the Wall Street Journal corpus) are as follows:

> As your editorial rightly pointed out, Samuel Pierce, former HUD secretary, and Lance Wilson, Mr. Pierce's former aide "are currently being held up to scorn for taking the Fifth Amendment."
>
> Thus, when Mr. Pierce asserted the Fifth in a noncriminal proceeding, particularly after presumably receiving extensive advice from legal counsel, one

System	Precision	Recall	F-score
Roark 2001 (CNF)	86.6	86.5	86.5
Described Model (CNF, beam width 500)	86.6	87.3	87.0
Described Model (CNF, beam width 2000)	87.8	87.8	87.8
Described Model (CNF, beam width 5000)	87.8	87.8	87.8
Petrov Klein (CNF)	88.1	87.8	88.0
Petrov Klein (not CNF)	88.3	88.6	88.5

Table 3.1: Accuracy comparison with state-of-the-art syntactic parsers. Numbers in parentheses are the number of parallel activated hypotheses. The left-corner parser used here restricts trees to Chomsky Normal Form (CNF), in which trees are binary branching at all nonterminals except preterminals. This makes the model less able to reproduce unary branches in the Penn Treebank.

must conclude that he held a good-faith, justifiable belief that his testimony could be used against him in a subsequent criminal prosecution.

I do not by any means defend HUD management.

Section 23 sentences contain an average of 20 words, leading to some quite complex sentences, as shown. The parsing results are shown in Table 3.1.

3.9 Discussion

The results presented in the previous subsection indicate that the hierarchic sequence model described can obtain similar accuracy to other state-of-the-art methods of parsing. The model has some unique properties in being incremental and also in its limitation of the grammar in ways that seem to mirror human grammar processing. Therefore, this model was used as the basis for further experiments.

Chapter 4

Speech Repair Model

This chapter describes extensions to the model described in Chapter 3 to accommodate speech repair. Section 4.1 describes the ways in which the probabilistic models must be altered to accommodate speech repairs. Section 4.2 describes ways in which the syntax trees are altered before training and testing. Finally, section 4.3 describes the results of parsing and edit detection using some of the changes described here.

4.1 Changes to the Incremental Parsing Model for Speech Repair

Beginning with the model described in Chapter 3, alterations are made so that the parser can posit nodes that are unfinished, (labeled as 'UNF' in the equations, and in highlight) and therefore part of a speech repair. The ϕ_d model predicts whether to introduce a new subordinate sequence, or not. Originally, it is split into the two possibilities F+ (*do* introduce a new sequence,) and F– (do not introduce a new sequence.) The ϕ_d model is altered by allowing for the possibility of predicting 'UNF' in addition to either introducing a subordinate sequence or not. In this case, the 'UNF' prediction

indicates that the current observation is part of a speech error. Specifically, the current observation is at the tail end of a speech error, and the current subordinate sequence may not terminate gracefully according to **L** rules. The reasoning is that speech errors often occur in the middle of a constituent, and at that point speech may be abruptly halted while the speaker attempts to figure out what the repair should be.

The probability of 'UNF' depends directly on the probability of 'UNF' nodes in the γ model, and its corresponding probability mass is taken from the probability of not introducing a subordinate sequence (F–).

$$\mathsf{P}_{\phi_d}(\text{`+'} \mid b\, p) \stackrel{\text{def}}{=} \frac{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{0} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)}{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{*} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)} \tag{4.1a}$$

$$\mathsf{P}_{\phi_d}(\text{`–'} \mid b\, p) \stackrel{\text{def}}{=} \frac{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)}{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{*} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)} - \frac{\mathsf{P}_{\gamma_{B,d}}(b{\to}p\ \text{`UNF'}) \cdot \sum_x \mathsf{P}_\gamma(p \to x)}{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{*} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)} \tag{4.1b}$$

$$\mathsf{P}_{\phi_d}(\text{`UNF'} \mid b\, p) \stackrel{\text{def}}{=} \frac{\mathsf{P}_{\gamma_{B,d}}(b{\to}p\ \text{`UNF'}) \cdot \sum_x \mathsf{P}_\gamma(p \to x)}{\mathsf{E}_{\gamma_d^*}(b \xrightarrow{*} p\ ...) \cdot \sum_x \mathsf{P}_\gamma(p \to x)} \tag{4.1c}$$

Next, models α_d and β_d must be altered. The original active event model α_d (defined in Equation 3.12) predicts the probability of an active event a, given an event b from which that active event is descended, and an event (called a'' in Equation 3.12, here called p.) Similarly, the original awaited event model β_d (defined in Equation 3.13) predicts the probability of an awaited event b' given that it is a right child of an event of category a' and that event has a left child of category a'.

Both α_d and β_d now must depend on the result of the ϕ_d prediction. In the equations, this is shown as added dependency f, which is the result of the prediction of the ϕ_d model.

If an 'UNF' is predicted, then these models calculate its probability by counting unary productions. The reason for counting only unary productions is that the only

place where unary productions occur is in speech repairs. This is an artificially generated situation that is a result of tree transformations on the training trees, which will be described in Section 4.2.

For the α_d model, if an 'UNF' is predicted, the probabilities are calculated from the expected counts of syntactic category a' occurring as an initial sub-event of category b multiplied by the probability of that event having a left child of category a'' multiplied by the probability of that left child a'' having only one child a''' (that is, being a unary production.) In equation 4.2, this is represented as having children a''' and '–':

$$\mathsf{P}_{\alpha_d}(a' \mid f\, b\, a''') \stackrel{\text{def}}{=} \begin{cases} \text{if } f \neq \text{'UNF'} : \dfrac{\sum_{b'''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ \ldots) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''' \ b''')}{\sum_{a',b'''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ \ldots) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''' \ b''')} \\ \text{if } f = \text{'UNF'} : \dfrac{\sum_{a'',b''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ \ldots) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a'' \ b'') \cdot \mathsf{P}_{\gamma_{A,d}}(a'' \rightarrow a''' \ \text{'–'})}{\sum_{a',b'''} \mathsf{E}_{\gamma_d^*}(b \xrightarrow{+} a' \ \ldots) \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a''' \ b''')} \end{cases} \tag{4.2}$$

The awaited event model $\beta_{s,d}$ is also altered if ϕ_d predicts 'UNF'. In that case, the $\beta_{s,d}$ probability is the probability that an event of category a has a right child of category b' given that it has a left child of category a' and that left child a' has only one child a'', again, meaning a unary production and being shown in Equation 4.3 as having children a'' and '–':

$$\mathsf{P}_{\beta_{s,d}}(b' \mid f\, a\, a'') \stackrel{\text{def}}{=} \begin{cases} \text{if } f \neq \text{'UNF'} : \dfrac{\mathsf{P}_{\gamma_{B,d}}(a \rightarrow a'' \ b')}{\sum_{b'} \mathsf{P}_{\gamma_{B,d}}(a \rightarrow a'' \ b')} \\ \text{if } f = \text{'UNF'} : \dfrac{\sum_{a'} \mathsf{P}_{\gamma_{B,d}}(a \rightarrow a' \ b') \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a'' \ \text{'–'})}{\sum_{a',b'} \mathsf{P}_{\gamma_{B,d}}(a \rightarrow a' \ b') \cdot \mathsf{P}_{\gamma_{A,d}}(a' \rightarrow a'' \ \text{'–'})} \end{cases} \tag{4.3}$$

Finally, the full probabilities for the σ model must be altered. The original σ model, described in Equation 3.5, contained four terms, which was the result of ϕ and λ predicting all four combinations for the two possible values ('+' and '–') for each model. Now that there is a third possible value for ϕ, the σ model will have six terms.

The differences from the original equation are shown in highlight. Again, $[\![\cdot]\!]$ is used as an indicator function, with value 0 or 1 depending the whether its argument is true.

Two new cases to handle $\phi =$ 'UNF' are added to the σ model, one for each of $\lambda_{=}$'+' and $\lambda_{=}$'–'.

$$
\begin{aligned}
&\mathsf{P}_{\sigma}(q_t^{1..D} \mid q_{t-1}^{1..D}\ x_{t-1}) \\
&\quad \stackrel{\text{def}}{=} \mathsf{P}_{\phi_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot [\![a_t^{d'-1} = a_{t-1}^{d'-1}]\!] \cdot \mathsf{P}_{\beta_{\mathrm{B},d'-1}}(b_t^{d'-1} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \\
&\qquad \cdot [\![q_t^{1..d'-2} = q_{t-1}^{1..d'-2}]\!] \cdot [\![q_t^{d'..D} = \text{`}-\text{'} ]\!] \cdot [\![ b_{t-1}^{d'-1} \neq \text{`UNF'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'}}(\text{`}-\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot \mathsf{P}_{\alpha_{d'}}(a_t^{d'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot \mathsf{P}_{\beta_{\mathrm{A},d'}}(b_t^{d'} \mid a_t^{d'}\ a_{t-1}^{d'+1}) \\
&\qquad \cdot [\![q_t^{1..d'-1} = q_{t-1}^{1..d'-1}]\!] \cdot [\![q_t^{d'+1..D} = \text{`}-\text{'} ]\!] \cdot [\![ b_{t-1}^{d'-1} \neq \text{`UNF'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot [\![a_t^{d'} = a_{t-1}^{d'}]\!] \cdot \mathsf{P}_{\beta_{\mathrm{B},d'}}(b_t^{d'} \mid b_{t-1}^{d'}\ x_{t-1}) \\
&\qquad \cdot [\![q_t^{1..d'-1} = q_{t-1}^{1..d'-1}]\!] \cdot [\![q_t^{d'+1..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`}+\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}-\text{'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\alpha_{d'+1}}(a_t^{d'+1} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\beta_{\mathrm{A},d'+1}}(b_t^{d'+1} \mid a_t^{d'+1}\ x_{t-1}) \\
&\qquad \cdot [\![q_t^{1..d'} = q_{t-1}^{1..d'}]\!] \cdot [\![q_t^{d'+2..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`UNF'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}+\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot [\![a_t^{d'-2} = a_{t-1}^{d'-2}]\!] \\
&\qquad \cdot \mathsf{P}_{\beta_{d'-1}}(\text{`UNF'} \mid \text{`UNF'}\ b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot [\![b_t^{d'-1} = b_{t-1}^{d'-1}]\!] \cdot [\![q_t^{1..d'-2} = q_{t-1}^{1..d'-2}]\!] \cdot [\![q_t^{d'..D} = \text{`}-\text{'}]\!] \\
&\quad + \mathsf{P}_{\phi_{d'}}(\text{`UNF'} \mid b_{t-1}^{d'}\ x_{t-1}) \cdot \mathsf{P}_{\lambda_{d'+1}}(\text{`}-\text{'} \mid b_{t-1}^{d'-1}\ a_{t-1}^{d'}) \cdot \mathsf{P}_{\alpha_{d'}}(a_t^{d'-1} \mid \text{`UNF'}\ b_{t-1}^{d'-2}\ a_{t-1}^{d'-1}) \\
&\qquad \cdot \mathsf{P}_{\beta_{d'}}(\text{`UNF'} \mid \text{`UNF'}\ a_{t-1}^{d'}\ a_{t-1}^{d'+1}) \cdot [\![q_t^{1..d'-2} = q_{t-1}^{1..d'-2}]\!] \cdot [\![q_t^{d'..D} = \text{`}-\text{'}]\!] \qquad (4.4)
\end{aligned}
$$

4.2 Tree Transformations

The corpus most often used for the task of speech-repair parsing is the switchboard corpus of spontaneous speech. The corpus has been annotated for training and testing, but other researchers (Johnson and Charniak, 2004, Hale et al., 2006, Miller, 2009) have found that the annotations that are available contain both too much information, and not enough information.

On one hand, the annotations contain punctuation, partial words, traces, and occasionally non-speech sounds, such as laughter. Including punctuation and traces is

problematic, as these annotations are (clearly) not in the speech signal, and the goal is to create a system that eventually works all the way from speech to grammatical structure. Therefore, our system does not use traces in its grammatical model, nor does it attempt to use punctuation information. Partial words are also annotated. These could be of use, as often a partial word precedes a speech repair, due to the speaker following Levelt's Main Interruption Rule. However, if input contains word fragments, an acoustic recognizer must use a model that recognizes that a pronunciation can end after any phone in a word's pronunciation, rather than only after certain strings of phones. The result is that a much more complicated acoustic system than currently exists would be needed to accurately recognize partial words. Therefore, the information about punctuation, non-speech sounds and traces is all discarded prior to training.

On the other hand, the annotations are extremely simplified in cases of speech repairs. When a speech repair is encountered, the entire reparandum is put inside of a constituent labeled "EDITED". Any unfinished constituents are given the additional tag "-UNF", then new constituents (the repair) are started without any special label. Other research (Miller, 2009, Hale et al., 2006) has shown that parsing results can be much improved by making deterministic tree transforms that add further information about the nature of the entire repair.

This section describes four different types of tree transformation that were tried in the system. First, unfinished constituents are dealt with differently. Then daughter annotation, which was used by Hale et al. (2006) and others with good results is applied. Interregnum labeling, a logical extension of daughter annotation, was attempted but ultimately discarded. Finally, top-level speech repairs were broken apart into multiple utterances.

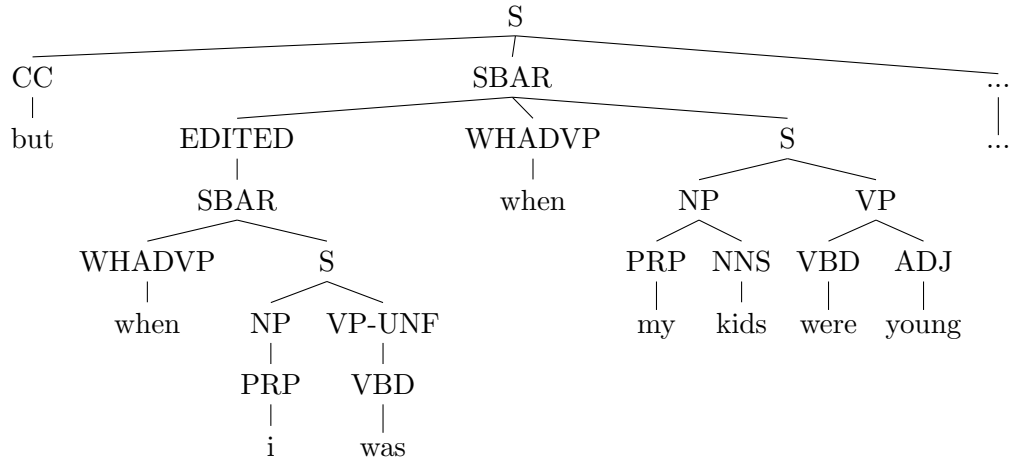

Figure 4.1: Part of a sentence from the Switchboard development set which will be used as an example in this chapter.

4.2.1 Unfinished "UNF" Consitutents

As mentioned previously, original switchboard annotation uses an addition of "-UNF" to a constituent's category label to indicate that the particular constituent is unfinished. This particular construction has two major disadvantages. First, the number of possible part-of-speech category labels is effectively doubled. Second, the tags that do have an "-UNF" marker tend to be less predictable, especially those that are higher-level constituents.

To get away from these disadvantages, the "-UNF" portion of each category label is removed, and a special node with head of "UNF" and unary production "!unf!" is added inside each of those constituents. Figure 4.2 shows an example of this transformation.

Then, "-UNF" nodes that do not occur inside of "EDITED" nodes are discarded prior to training. This makes sense, as if the speaker did not feel that there was any need to repair the utterance, then that speaker must have considered the utterance to be grammatical (or, grammatical enough to make their meaning understood). Therefore, there is not any point to our system treating a constituent as "unfinished" when the

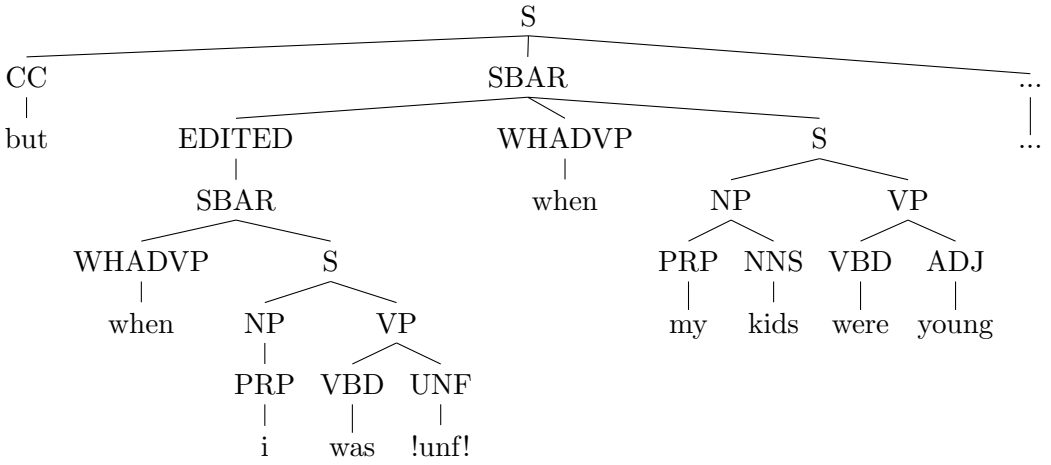

Figure 4.2: The sentence from Figure 4.1, after dealing with "-UNF" constituents.

meaning was conveyed and the speaker considered the utterance to be grammatical.

4.2.2 Daughter and Sibling Annotation

A single "EDITED" constituent for all speech repairs leads to difficulties in matching reparanda with their repairs. Hale et al. (2006) solved this problem using "daughter annotation" which added a tag to "EDITED" labels to match their children. Thus, "EDITED" will become "EDITEDNP" if NP is its first child.

Another difficulty with the original annotation is that it requires non-local operations to classify which constituents are part of a repair. This model uses a sibling annotation process in which siblings of "EDITEDx" nodes are placed under a new constituent with the label "REPAIRx". The "REPAIRx" constituent then covers the repair and all editing terms. Figure 4.3 shows an example of the tree from Figure 4.2 after these transformations.

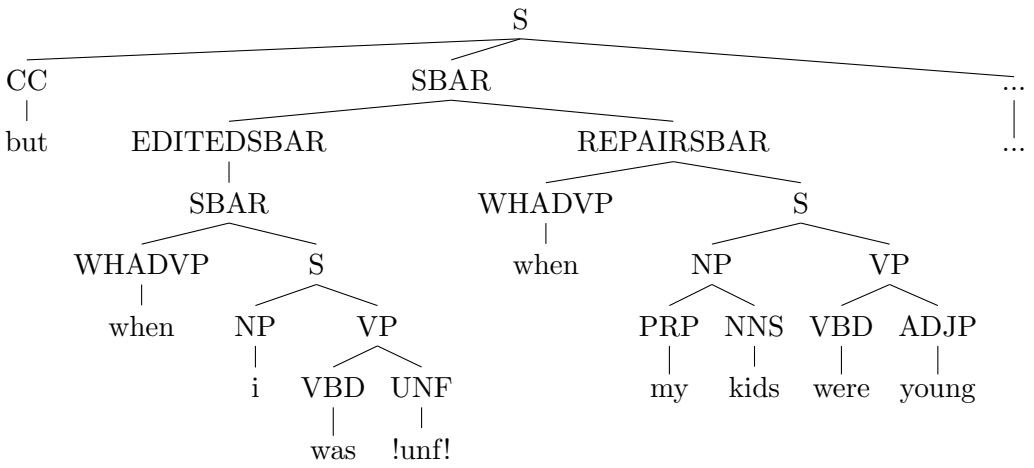

Figure 4.3: The same sentence from the Figure 4.2, after undergoing daughter and sibling annotation.

4.2.3 Segmentation at the Sentence Level

Certain tree geometries involving speech repairs present problems for the parser. When the parser generates an"UNF", it copies forward constituents and attempts to transition to the next higher depth level in the q model. The assumption is that the current connected component should be treated as "completed". However, when a repair occurs at the highest level of an utterance, there are no components in the higher depth levels in the q states, so such parses wind up with a probability of 0.

To solve the problem, trees with top-level speech repairs were segmented into separate utterances. Figure 4.4 shows a tree with a top-level repair that would be split into two separate utterances.

4.2.4 Interregnum Labeling

In the labeling described in section 4.2.2, all speech repairs are broken apart into EDITEDx and REPAIRx constituents. In general, the EDITEDx constituents consist of a partial or complete x constituent, up to the point where the speaker realized

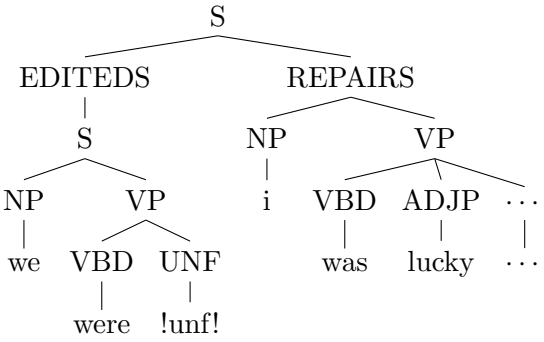

Figure 4.4: A portion of a sentence from the training set with a top-level repair that was segmented into multiple utterances.

an error. The rest of the repair is contained within the REPAIRx constituent.

Therefore, an EDITEDx constituent will consistently be similar to an x constituent (although it may be missing pieces,) but a REPAIRx may contain any number or type of filled pauses, because it necessarily must contain the rest of the speech repair. With interregnum labeling, a special tag INTERx is given to the interregnum when it exists. This way, a REPAIRx constituent should also consistently be similar to an x constituent, and more importantly it will be structurally similar to other REPAIRx constituents, allowing the model to be more specific about what a REPAIRx node looks like. An example of interregnum labeling is shown in Figure 4.5.

Unfortunately, this method did not improve parsing results, in fact it hurt them significantly. Results on the development set resulted in a loss of around 4% parsing score, so this alteration was never run on the test set.

This result was surprising at first, as this tree relabeling seemed to be a straightforward process for which it would be easy to generate predictions. However, not all repairs actually have an interregnum, therefore there was likely a data sparsity issue. Furthermore, the model did not actually gain any benefit from a REPAIRx constituent having structural similarity to a non-repair x constituent, as it did not have any notion

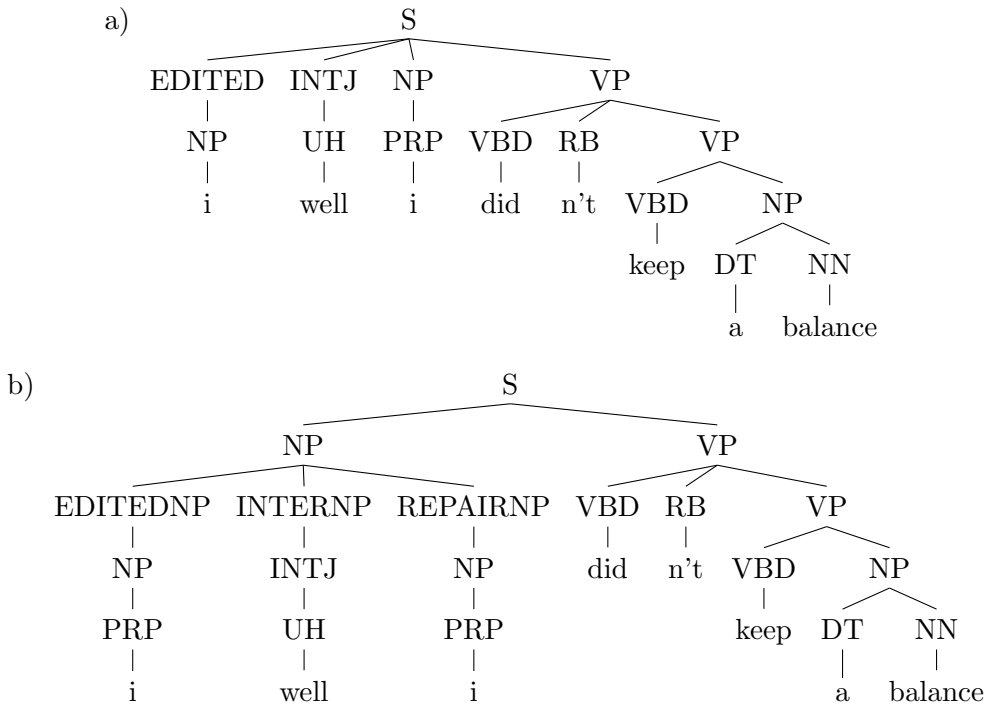

Figure 4.5: Part of a sentence from the Switchboard development set, as it is originally (a), and then after tree transformation including interregnum labeling (b).

of those being categories that should be related. Therefore, this transform just adds a new constituent without gaining much informational benefit.

4.3 Results

4.3.1 Calculation of Edit-Detection

To calculate results of edit detection, each word in a sentence is labeled with a 0 or 1, depending on whether it has an EDITED constituent ancestor or not. Then, accuracy and precision are calculated on a word-by-word basis. From these measures, F-Score is calculated as the harmonic mean of precision and recall.

To calculate precision and recall, we measure the number of true positives *tp* (words

that were predicted as inside an EDITED that were inside an EDITED in the original treebank), false positives *fp* (words that were predicted as inside an EDITED that were not inside an EDITED in the original treebank), true negatives *tn* (words that were predicted as not inside an EDITED that were not inside an EDITED in the original treebank), and false negatives *fn* (words that were predicted as not inside an EDITED that were inside an EDITED in the original treebank).

$$Precision = \frac{tp}{tp + fp} \tag{4.5}$$

$$Recall = \frac{tp}{tp + fn} \tag{4.6}$$

$$F\text{-}Score = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall} \tag{4.7}$$

4.3.2 Comparison

This system is an improvement upon previous incremental parsers with speech repair handling. Table 4.1 shows the comparison with other state-of-the-art speech repair parsers. This system performs worse than non-incremental parsers, and worse than specialized edit-detection systems, but for the particular task of incremental, single-pass, parsing and speech repair detection, this system is an improvement. That particular task may seem narrow, but a system that is attempting to interact with humans in a conversational setting will likely want an incremental approach (so that it can generate hypotheses as it hears each word of speech) and be able to deal with both parsing and edit detection.

The results shown in the table seem to differ drastically from Chapter 3, but the results in Chapter 3 are on the Penn Treebank corpus, which does not contain speech repairs. The standard corpus for testing speech repair is the Switchboard corpus, and those results are shown in the table.

System	Edit-F	Parse-F
Miller (2009), incremental single-pass	53.6	74.2
Hale et al. (2006) baseline, non-incremental	57.6	83.86
Described model, incremental single-pass	61.9	81.7
Zwarts et al. (2010), incremental multi-pass	77.8	-
Johnson and Charniak (2004), non-incremental	79.7	-

Table 4.1: F-Score comparison with other state-of-the-art speech repair parsers on the Switchboard corpus.

Chapter 5

Improvements Specific to Speech Repair

Chapter 4 described modifications that were necessary for a domain-general hierarchic sequence prediction model (described in Chapter 3) to be able to handle specific aspects of speech repairs. The result of those modifications was a system that was better than previous systems, but not competitive with other systems that specialize in edit detection or parsing only.

This chapter contains descriptions and results of a buffer model subsystem that was added to the parser to specifically improve speech repair handling. Section 5.1 describes the theory behind the buffer model. Section 5.2 describes the implementation in this particular system. Section 5.3 summarizes the results for various systems.

Miller (2009), further expanded in Miller (2010), added a buffer to the system that is intended to model phonological short-term memory. Shriberg (1994) shows that most speech disfluencies are deletions of 2 or fewer words, and in fact the occurrences of disfluencies fall off exponentially as the number of words in the disfluency increases.

5.1 Buffer Model

Baddeley and Hitch (1974) hypothesizes a "phonological loop" (sometimes referred to as an "articulatory loop") in human memory that maintains recently encountered words in a particular space in short-term memory. There are two parts of this phonological loop, a short-term phonological store that keeps auditory memory traces that decay rapidly, and the articulatory rehearsal component, that can access the memory traces of the store. This memory is accessed during human speech production for subvocal rehearsal of speech before actual articulation. If a speech disfluency occurs, the most recently spoken words would still be within the phonological store. Therefore, if this phonological loop is involved during speech disfluency repairs, it would suggest that speakers are more likely to repeat some of the words in the reparandum, due to its existence in the phonological store, rather than coming up with entirely new strings of words to replace what was deleted in a repair.

Using this theory of a phonological loop, a buffer can be added to the model to represent this short-term memory that will influence the probability of generating a parse with a speech repair component when certain sequences of repetitions are encountered. The buffer would store previously spoken words, and when a repeat of one of the words in the buffer is encountered, this would influence the probability that a repair is in progress. This buffer could be further specialized by the observation that editing terms *um*, *uh*, and so on, are not stored in the phonological loop, and may be safely ignored, essentially giving the buffer a larger memory in cases where these terms are used.

5.2 Buffer Implementation

A previous implementation Miller (2010) of a buffer model improved detection of speech repairs in a different parser. Miller (2010) and Shriberg (1994) indicate that a 4-word

buffer is sufficient to handle 96% of speech disfluencies that appear in the Switchboard corpus.

The system, as described previously, is in the following configuration:

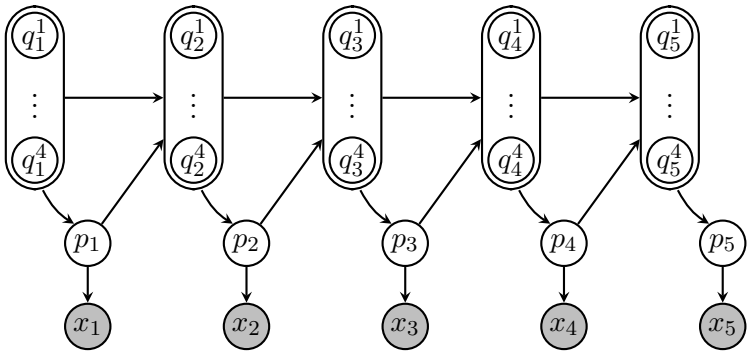

Figure 5.1: The current system

The buffer model adds two sub-models to system, an interruption point (i) variable, and a word buffer (c) model, and alters the x model. These random variables are then inserted into the model, with the i model depending on the q and f variables, and the c model depending on the q and i variables, and also generating the observations. Figure 5.2 shows how the model looks with these two random variables added.

The I variable has a domain of {**0, 1, ET**}, where **1** indicates the first word of a repair, **ET** indicates an editing term between reparandum and repair, and **0** indicates no repair. In all the experiments, the I model was deterministic, as Miller (2010) showed that such a model could contribute positively. Different configurations of this deterministic model were used in different experiments.

The random variable for the word buffer is a non-deterministic model with the domain {**0, 1, ...,** n} for buffer size n. The random variable indicates the current position in the buffer as a "rewind" amount, i.e. how far back into the buffer the system is looking.

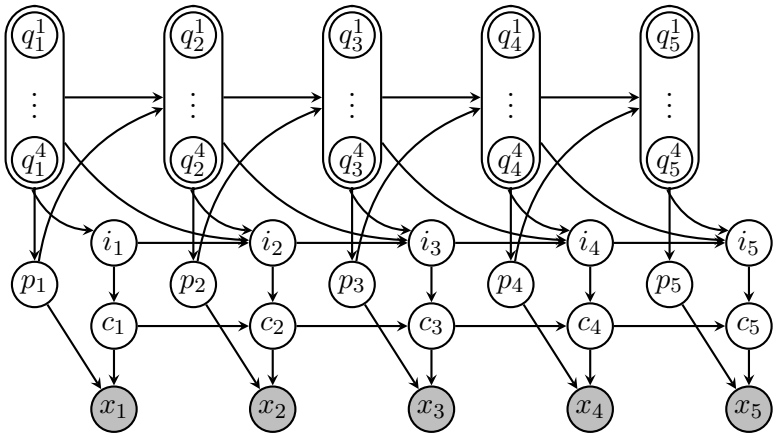

Figure 5.2: A diagram of the model with buffer and interruption point models integrated.

5.2.1 The Interruption Point Variable

Miller (2010) posits a deterministic Interruption Point variable that depends on predicted part-of-speech constituents in its parser. Specifically, it relies on predictions of **INTJ** and **PRN** constituents to determine whether a word is an editing term, and completed **EDITED** constituents to determine whether a word is an Interruption Point.

The topology of this system required a slight change to the interruption point model, as follows. the Interruption Point variable depends on predictions of the previous and current time step's active and awaited variables. Predictions of unfinished constituents ('UNF',) filled pauses ('FP',) and repairs ('REP') have an effect on the variable. The notation $\in$ is used here to mean that a particular constituent type appears in a random variable or a part of a random variable.

Two different models were used. The difference between the two models is that P_{I_2} integrates information about 'REP' appearing in the current timestep's awaited constituent, whereas P_{I_1} does not. The models here are short-circuited, such that proceeding from top to bottom, the first true condition encountered produces the interruption

point variable's value.

$$\mathsf{P}_{I_1}(i_t \mid i_{t-1}, q_t, q_{t-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } UNF \in q_{t-1} \wedge FP \in q_t & : [i_t = ET] \\ \text{if } UNF \in q_{t-1} \wedge FP \notin q_t & : [i_t = 1] \\ \text{if } i_{t-1} = ET \wedge FP \in q_t & : [i_t = ET] \\ \text{if } i_{t-1} = ET \wedge FP \notin q_t & : [i_t = 1] \\ \text{if } i_{t-1} = 0 & : [i_t = 0] \end{cases} \tag{5.1}$$

$$\mathsf{P}_{I_2}(i_t \mid i_{t-1}, q_t, q_{t-1}) \stackrel{\text{def}}{=} \begin{cases} \text{if } (UNF \in q_{t-1} \vee REP \in b_t) \wedge FP \in q_t & : [i_t = ET] \\ \text{if } (UNF \in q_{t-1} \vee REP \in b_t) \wedge FP \notin q_t & : [i_t = 1] \\ \text{if } i_{t-1} = ET \wedge FP \in q_t & : [i_t = ET] \\ \text{if } i_{t-1} = ET \wedge FP \notin q_t & : [i_t = 1] \\ \text{if } i_{t-1} = 0 & : [i_t = 0] \end{cases} \tag{5.2}$$

5.2.2 The Buffer Rewind Variable

The operation of the buffer is governed by 4 distinct cases:

Case 1: During fluent speech, the I variable is at 0 and the buffer rewind amount at the previous time step c_{t-1} is 0. In this case, the last $n-1$ items in the buffer are copied back 1 position, and a new word is probabilistically generated to occupy the last position in the buffer. This probability is estimated empirically using the model as described in Chapter 4.

Case 2: When an editing term is being generated, $i_t = \mathbf{ET}$, the buffer is not in use, all values are copied from the previous time step $t-1$ to the current time step t.

Case 3: The alteration case applies to the first word after the sequence containing the reparandum and optional editing terms, $i_t = \mathbf{1}$. In this case, c_t is obtained by determining the amount to rewind. Two different models were used in experiments, one based off Miller (2010) and Shriberg (1994), (designated R_{OLD}) and another based

on my own analysis of speech errors and repairs in the Switchboard corpus (designated R_{TRAIN}). My analysis was based on a count of lengths of $EDITED$ constituents in the training set. It differs from R_{OLD} in that R_{OLD} was trained on multiple corpora.

Case 4: When the buffer had been activated in a previous time step (i.e. c_{t-1} is not 0,) then the buffer model is used to influence the probability of generating the next word, as described in the next section. The buffer values are copied forward from the previous time step and a new rewind amount c_t is probabilistically generated. Different configurations were used for this case. One is a deterministic model, FR_{DET} which forces the buffer rewind amount to decrease by 1 at each time step. Another version allows the buffer rewind amount to stay the same or even skip forward with a low probability, allowing the buffer model to help even if words were added or deleted from the repair. This model is designated FR_{PROB}.

5.2.3 Modifications to the Observation Model

The observation model ξ was altered to depend not only on the predicted preterminal, but also on the buffer variables i and c.

The model is then defined as follows:

$$\mathsf{P}_\xi(x \,|\, p, i_t, c_t) \stackrel{\text{def}}{=} \begin{cases} \text{if } c_t \neq 0 \wedge BUF[c_t] = x_t & : \lambda_{buf} \cdot pr_{buf} + (1 - \lambda_{buf}) \cdot \mathsf{P}_\xi(x \,|\, p) \\ \text{if } c_t \neq 0 \wedge BUF[c_t] \neq x_t & : \lambda_{buf} \cdot pr_{sub} + (1 - \lambda_{buf}) \cdot \mathsf{P}_\xi(x \,|\, p) \\ \text{else} & : \mathsf{P}_\xi(x \,|\, p) \end{cases} \tag{5.3}$$

That is, if the buffer is on, and word in the buffer at the rewind amount $BUF[c_t]$ matches the observed word, then a weighted average (controlled by λ_{buf}) between the original probabilistic model and the buffer match probability pr_{buf} is returned. If the buffer is on but the word in the buffer at the rewind amount does not match the observed

word, then a substitution probability pr_{sub} is multiplied by the original probability. In all other cases, the original probabilistic model is used.

5.3 Experiments and Results

5.3.1 System Parameters

The system was trained on sections 2 and 3 of the Switchboard corpus (the standard training set for parsing when using Switchboard.)

Results for the system are given as F-Scores in Edit Detection. Edit detection results are measured in the following manner: Each word is tagged with a boolean value of being within an **EDITED** or not. Results are then measured on a word-by-word basis to produce precision, accuracy and F-Score. The output trees shown in this chapter are the result of a deterministic modification of the parser's output to remove REPAIR constituents and flattening binary trees. These modifications allow for an easier comparison with the gold standard. The different variables tested for the model are as follows:

1. Buffer influence weight amount λ_{buf}
2. Choice of interruption point model, P_{I_1} or P_{I_2}
3. Rewind amount model, R_{OLD} or R_{TRAIN}
4. Deterministic rewind model or not, FR_{DET} or FR_{PROB}
5. Substitution probability pr_{sub}

5.3.2 Buffer Weight

Tests were performed to see whether there was a particular maximal λ_{buf} to set. Using P_{I_1}, FR_{DET}, R_{TRAIN} and $pr_{sub} = 0.1$, values from 0 (no buffer model influence) to

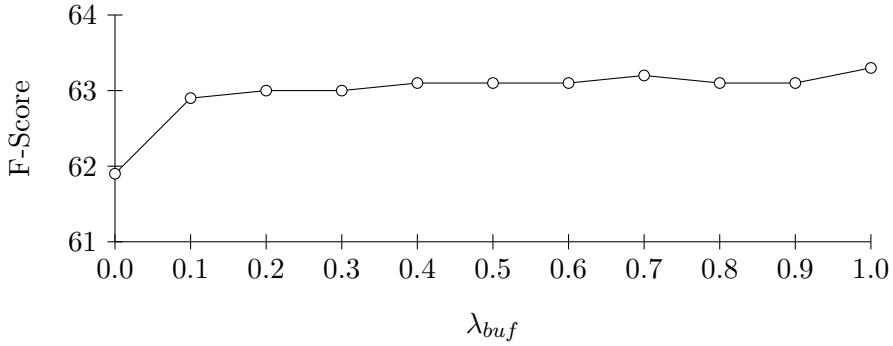

Figure 5.3: P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.1$

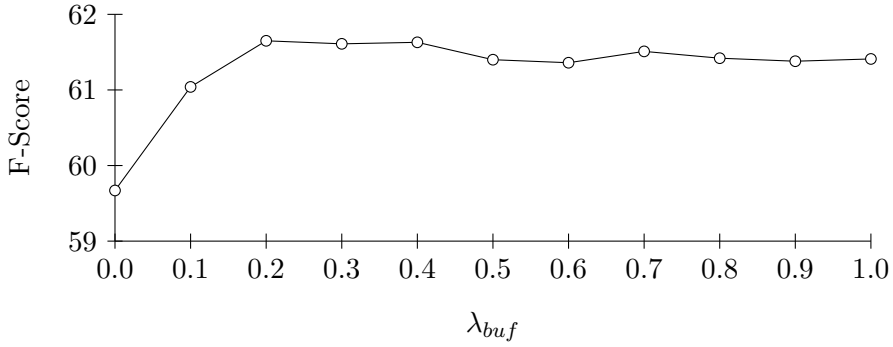

Figure 5.4: P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.1$

1 in increments of .1 were tested. The results, shown in Figure 5.3, indicate that the the buffer is helpful. A first glance at the graph suggests that $\lambda_{buf} = 1.0$ may be ideal, however the differences between the results are not significant enough to draw a conclusion. A second test, with P_{I_2}, (results shown in Figure 5.4,) shows that there are some conditions in which $\lambda_{buf} = 1.0$ will not produce the best results.

For the remainder of the tests, buffer weights of 0.2, 0.4, 0.6, 0.8 and 1.0 were used.

5.3.3 Substitution Probability and Interruption Point Model

Further tests showed that interruption point model P_{I_1} performed better across all weights and substitution probabilities. This is surprising, as P_{I_1} tends to mis-predict

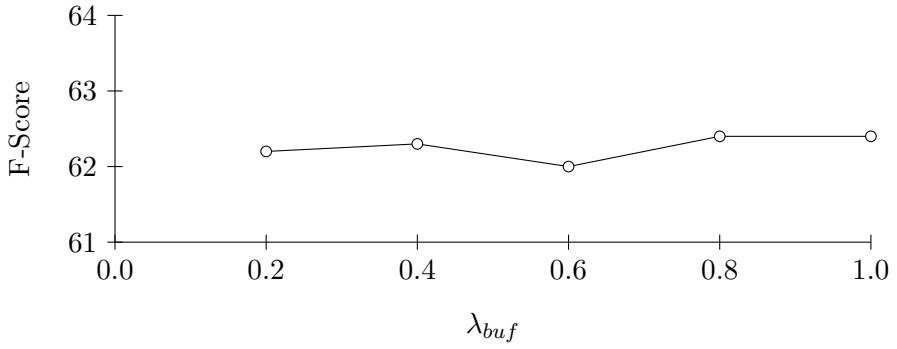

Figure 5.5: P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.5$

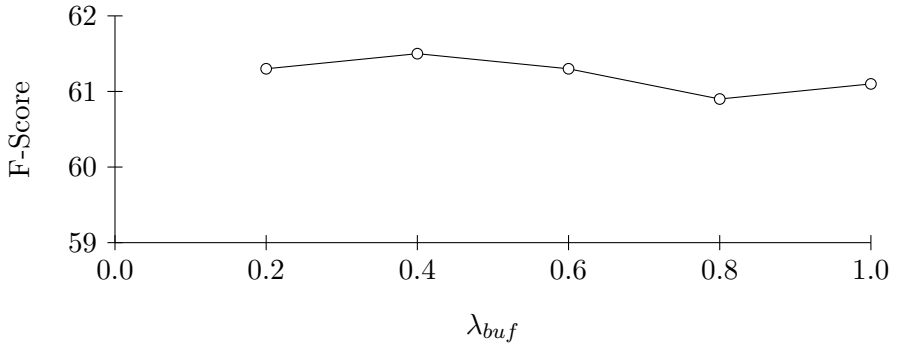

Figure 5.6: P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.5$

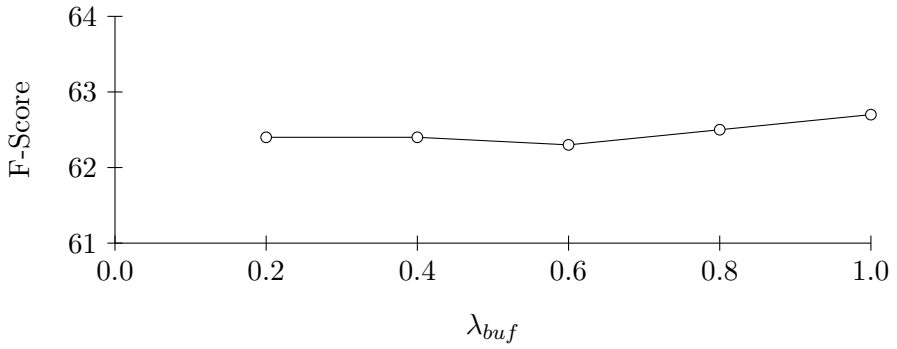

Figure 5.7: P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.9$

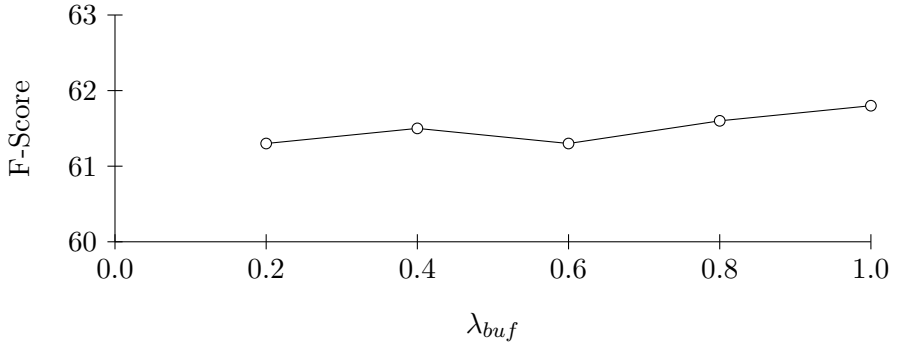

Figure 5.8: P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.9$

for simple speech repairs where a single word is repeated, since the training trees do not create an *UNF* constituent in these simple cases. Figure 5.9 shows an example of this sort of error. The system using P_{I_1} does not properly predict the EDITED constituent as a single word, as the parser would not have generated an UNF at the proper timestep. The system using P_{I_2} is able to correctly predict the EDITED constituent as the parser generates a REPAIR constituent as a possibility at the proper timestep.

A more detailed analysis showed that P_{I_2} has higher precision, but lower recall. This is a bit strange, as P_{I_2} allows for more cases of predicting speech errors, thus presumably allowing for more false positive, and a lower precision, but this is not the case. Furthermore, a lower recall also is surprising for the same reason. Another example of differences between results is shown in Figure 5.10. P_{I_2} prefers a flat tree with no EDITED constituents, in this case.

For the remainder of the tests, P_{I_1} and $pr_{sub} = 0.9$ were used.

5.3.4 Rewind Amount Model

The R model dictates probabilities for rewind amounts. When the system detects an error, it guesses how far back to rewind the buffer based on probabilities in R. Such a model can be generate by simply performing a count of lengths of all speech errors

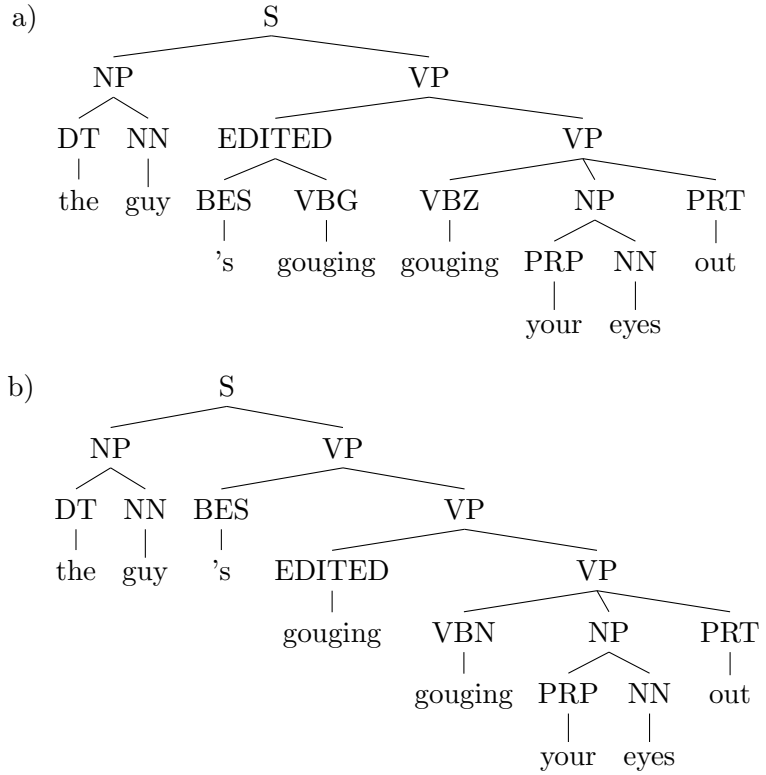

Figure 5.9: Output for sentence 14 from the test set from a system using P_{I_1} (a), and then the same tree from a system using P_{I_2} (b).

and normalizing. Shriberg (1994) generated such a model based on multiple coropora, and this model was subsequently used in Miller (2010) for positive effect. This model is designated R_{OLD} here. Another model was generated solely from the Switchboard corpus training set, and is designated R_{TRAIN} here.

5.3.5 Buffer Advance Model

Once a rewind has occurred, the system uses the buffer to generate words in the repair. If the words are simply repeats, then the system will give a high probability to generating those repeated words. However, while many speech errors are simple repeats, others

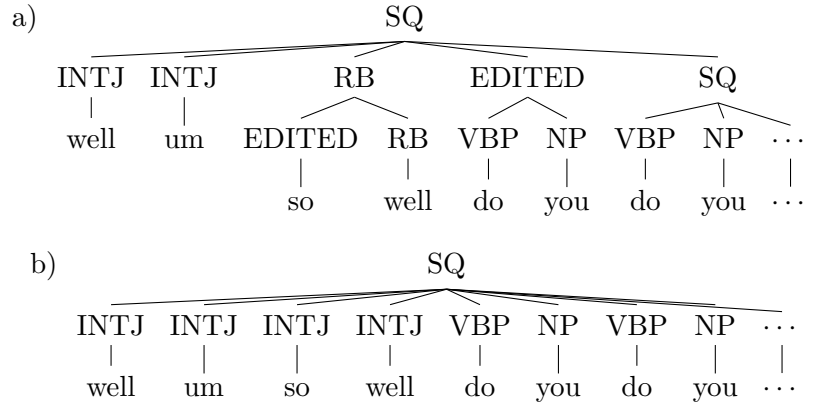

Figure 5.10: Partial output for sentence 37 from the test set from a system using P_{I_1} (a), and then the same segment from a system using P_{I_2} (b). The correct parse has only one EDITED constituent, over the first "do you".

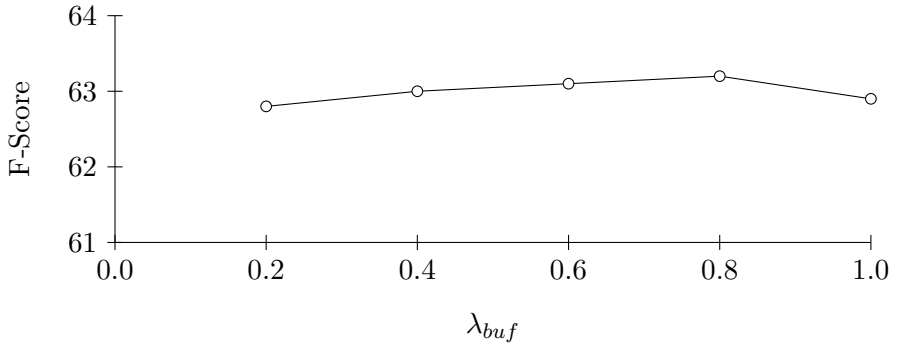

Figure 5.11: P_{I_1}, R_{OLD}, FR_{DET}, $pr_{sub} = 0.9$

include added words, and occasionally words are elided that were originally in the repair.

An example of this is in a sentence from the development set, *... I noted that the monthly salary starting average monthly salary ...* This sentence is shown with its syntactic structure in figure 5.13. Importantly, this particular sentence has an error with two words that is then fixed with a 4-word repair. As shown in the figure, a system using FR_{DET} fails to posit a repair for this structure. It cannot use the buffer system to model this, because the repair length is different than the error length.

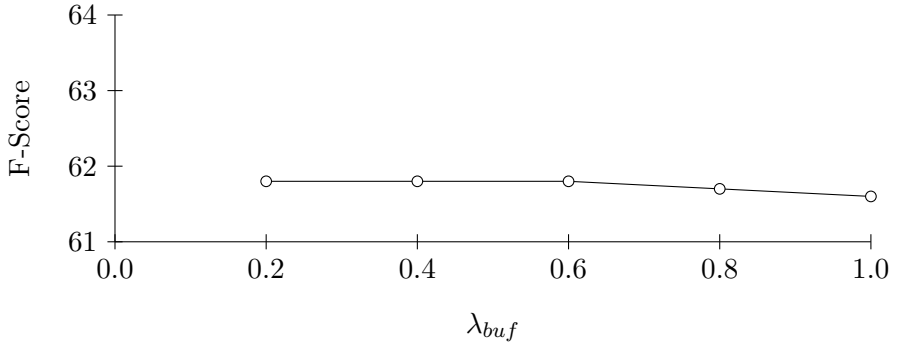

Figure 5.12: P_{I_1}, R_{TRAIN}, FR_{NONDET}, $pr_{sub} = 0.9$

The nondeterministic model was determined using a method similar to that of Miller (2010), who followed Johnson and Charniak (2004). Using the disfluency-annotated corpus, a list of reparanda and corresponding repairs are compiled. Then, the alignment cost is computed, where copy operations have cost 0, substitution operations have a cost 4, and insertion and deletion have cost 7. Using the alignments, probabilities are computed regarding the buffer activities from its current index. Results with this model, FR_{NONDET} are shown in Figure 5.12.

5.4 Discussion

The overall result of adding this system is a slight increase in edit-detection F-score. In some cases, the difference is not statistically significant. A closer analysis of the recall and precision scores can give us some results that are statistically significant, but essentially any gains in one metric are canceled out by losses in the other metric, as shown in Table 5.1.

The addition of this subsystem drastically helped (Miller, 2010), but only has a slight positive effect on this model. Our system is starting at a higher baseline than (Miller, 2010), and one possibility is that the improvements to the system that allow

System	F-Score	Precision	Accuracy
Baseline	61.9	76.9	51.8
P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.1$	63.3	73.9	55.4
P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.1$	61.7	83.1	49.0
P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.5$	62.4	70.8	55.8
P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.5$	61.5	77.9	50.8
P_{I_1}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.9$	62.7	70.4	56.6
P_{I_2}, R_{TRAIN}, FR_{DET}, $pr_{sub} = 0.9$	61.8	75.5	52.3
P_{I_1}, R_{OLD}, FR_{DET}, $pr_{sub} = 0.9$	63.2	69.4	58.0
P_{I_1}, R_{TRAIN}, FR_{NONDET}, $pr_{sub} = 0.9$	61.8	77.0	51.7

Table 5.1: Comparison of F-Score, Precision and Accuracy for various configurations of the buffer. For each configuration, the λ_{buf} that produces the best F-Score is shown here.

for speech repair result in enough gains that the buffer model cannot improve much. One way to investigate this would be to examine simple edits and repairs that would be caused by single-word repetitions. The model is set up to best account for repetitions, as it is attempting to increase probabilities of repairs when words and structure are repeated.

Another idea could be that this system would work better if more information than just words were saved by the buffer. Often speech repairs result in repetitions of entire structures, not just words, and if constituent information were incorporated into the buffer model, it may do a better job of predicting edit and repair constituents. A preliminary investigation of trying to incorporate constituent tags into the buffer was undertaken, essentially allowing the buffer to match constituent categories instead of exact words, but this change resulted a such a high level of false positives that it was not pursued.

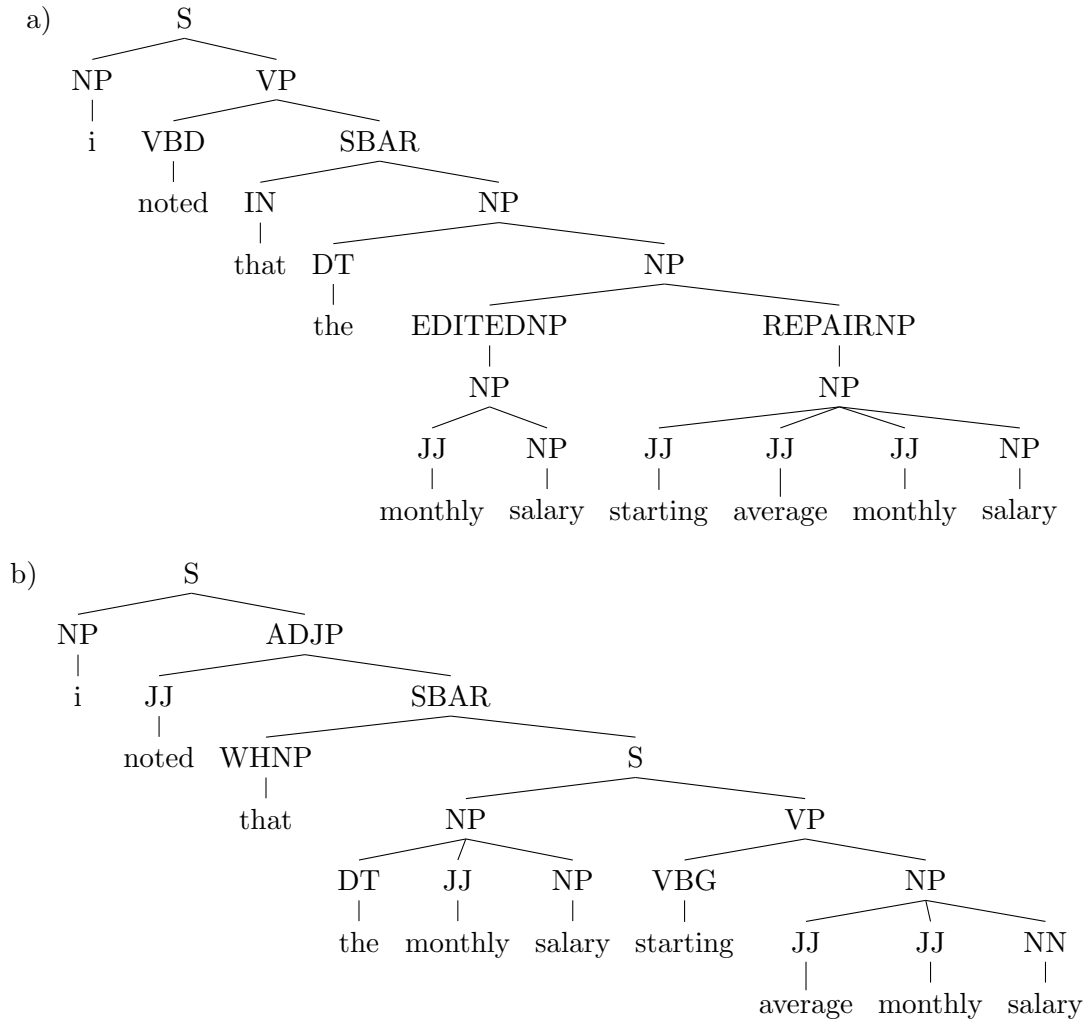

Figure 5.13: Portion of a sentence from the development set, (a) and the corresponding output parse from a system using FR_{DET} (b).

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis presents a competitive parsing system that is based on computational models of human memory and sequence understanding.

Chapter 3 introduces a connected component model of human language understanding and processing that uses ideas from models of other types of human sequence processing. In particular, it uses the idea that humans actively expect and predict future items and goal states given a sequence. The chapter describes this model as a general event sequence prediction framework, then describes how the framework may be applied to natural language parsing.

The chapter then examines the rules that are necessary to derive probabilistic models, and examines how to translate a context-free grammar into one that can be used by the system. This step is important, since much research has gone into training context-free grammars on various corpora. By showing that we can translate a given context-free grammar to one that is understood by the connected component system, we can then use another system's context-free grammar trainers to build models, then

translate them to a connected component grammar.

Finally, the chapter gives results that are of similar accuracy to other state-of-the-art syntactic parsers. The conclusion was that although the system is not strictly better than other syntactic parsers, it has some unique properties, in that it is incremental, and also that it limits the grammar in ways that are similar to human language processing.

Chapter 4 describes changes to the model to allow for handling of speech repairs. To do this, the model must be altered so that the expansion rule can handle a third possibility – that there was a speech error and that the utterance is about to be corrected. Adding this possibility requires changes to some of the model probabilities. Importantly, this change does not require any drastic topological changes to the model. Once the new model probabilities are calculated, the same system as in the previous chapter can be used to calculate the most likely parse for a given sentence.

This chapter also describes the tree transformations that are necessary to be able to train and test the system on the Switchboard corpus. Previous work has shown that daughter annotation is a simple deterministic alteration that results in improvements in test results, so that was incorporated into the system. Additionally, other alterations were tested and implemented.

The first new alteration was cleaning up the unfinished constituent structure. This resulted in reducing the number of constituents in the data, and instead adding new "UNF" nodes that indicate an unfinished constituent. Next was sibling annotation, where each EDITEDx constituent was forced to be a sibling of a REPAIRx constituent, which allows the system to train that, for example, an EDITEDNP will have a different structure than an EDITEDVP. These additional alterations also helped the system.

Another alteration that was tested was as an additional tag for interregnums, but this was not found to be helpful to the system.

The changes resulted in an improvement of both parsing and edit-detection on the

baseline system for the task of parsing on the Switchboard corpus. The results are not as good as specialized systems for edit-detection, nor is the parsing accuracy as good as other state-of-the-art parsers, but the parsing results are close.

Chapter 5 describes multiple experiments using a buffer system that models a human memory store that was able to improve results of a previous incremental system. The buffer model is essentially a short-term memory of content words that allows the system to "prefer" speech repairs if words are being repeated within a small time frame. The buffer was tested as it had been set up in the previous system, but it only produced a small improvement in edit-detection.

Different configurations of the buffer system were tested, including different models, and differing amount of penalties for non-matching words when a speech repair was generated, but none of them resulted in significant improvements to the system.

Next, the discrepancy between gains from the buffer system was analyzed. It had helped a previous incremental system, but not this one. The system described here has a higher parsing accuracy and edit detection accuracy than the previous system, before adding in the buffer model, and it seems likely that many of the gains afforded by the buffer model on the previous system are already accounted for by the current system. Therefore, the buffer model may not be adding much value to the system.

6.2 Future Work

While the system does not match state-of-the-art systems in terms of edit detection and parsing on sentences with speech repairs, it is still useful for certain applications. Its incremental nature allows for a system that generates best hypotheses as the system runs, which can allow for real-time interaction.

The system can serve as a base for applications in which real-time results are needed during an utterance, or as a base for systems that want to extend parsing by adding a

higher-level cognitive domain (for example, semantics,) so that the syntax hypotheses at each time step can be used to influence the most likely sequence calculations for higher levels.

There are still some areas in which improvements could be made. On one of the sample runs with the buffer, (specifically with P_{I_1}, R_{TRAIN}, FR_{NONDET}, $pr_{sub} = 0.9$), an analysis of sentences in which incorrect edit detection occurred showed that these sentences have an average length of 20.3 words. The average length of an edit in these sentences is 1.7 words. Breaking it down even further, precision errors occurred in 147 sentences, with an average length of 22.8 words, and an average edit length of 2.1 words. Recall errors occurred in 359 sentences with an average length of 21.8 words, with an average edit length of 1.3 words. The obvious point of improvement would be a focus on recall errors, which are more numerous. The buffer model was intended to bias the system towards more predictions of short, one- or two-word edits, but it did not help very much in those cases. It is possible that the buffer system needed more information in order to better increase probability of edits.

The system could incorporate syntactic information, increasing the probability of edits when parts of speech matched, rather than just exact words. As mentioned in Chapter 5, a brief investigation into this idea was undertaken, but there were too many false positives. Because of that, the idea was abandoned early, but it is possible that further research into this modification of the system would give better results.

Bibliography

Arnold, J. E., Fagnano, M., and Tanenhaus, M. K. (2003). Disfluencies signal theee, um, new information. Journal of psycholinguistic research, 32(1):25–36.

Atkinson, R. C. and Shiffrin, R. M. (1968). Human memory: A proposed system and its control processes. The psychology of learning and motivation, 2:89–195.

Baddeley, A. D. and Hitch, G. J. (1974). Working memory. The psychology of learning and motivation, 8:47–89.

Bellman, R. (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.

Botvinick, M. and Plaut, D. C. (2004). Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action. Psychological review, 111(2):395.

Botvinick, M. M. (2007). Multilevel structure in behaviour and in the brain: a model of fuster's hierarchy. Philosophical Transactions of the Royal Society B: Biological Sciences, 362(1485):1615–1626.

Clark, H. H. and Fox Tree, J. E. (2002). Using uh and um in spontaneous speaking.

Clark, H. H. and Wasow, T. (1998). Repeating words in spontaneous speech. Cognitive psychology, 37(3):201–242.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97).

Cowan, N. (1999). An embedded-processes model of working memory. Models of working memory: Mechanisms of active maintenance and executive control, 20:506.

Ericsson, K. A. and Kintsch, W. (1995). Long-term working memory. Psychological review, 102(2):211.

Freud, S. (1901). Psychopathology of Everyday Life. Macmillan Company.

Fuster, J. M. (1990). Prefrontal cortex and the bridging of temporal gaps in the perception-action cycle. Annals of the New York Academy of Sciences, 608(1):318–336.

Fuster, J. M. (1997). Network memory. Trends in neurosciences, 20(10):451–459.

Fuster, J. M. (2001). The prefrontal cortexan update: time is of the essence. Neuron, 30(2):319–333.

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). Switchboard: telephone speech corpus for research and development. In Proceedings of the 1992 IEEE international conference on Acoustics, speech and signal processing - Volume 1, ICASSP'92, pages 517–520, Washington, DC, USA. IEEE Computer Society.

Hale, J. (2001). A probabilistic earley parser as a psycholinguistic model. In Proceedings of the second meeting of the North American chapter of the Association for Computational Linguistics, pages 159–166, Pittsburgh, PA.

Hale, J. (2003). Grammar, Uncertainty and Sentence Processing. PhD thesis, Cognitive Science, The Johns Hopkins University.

Hale, J. (2006). Uncertainty about the rest of the sentence. Cognitive Science, 30(4):609–642.

Hale, J., Shafran, I., Yung, L., Dorr, B., Harper, M., Krasnyanskaya, A., Lease, M., Liu, Y., Roar, B., Snover, M., and Stewart, R. (2006). PCFGs with syntactic and prosodic indicators of speech repairs. In Proceedings of the 44th Annual Conference of the Association for Computational Linguistics (COLING-ACL).

Johnson, M. and Charniak, E. (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.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. Cognitive Science: A Multidisciplinary Journal, 20(2):137–194.

Levelt, W. J. (1983). Monitoring and self-repair in speech. Cognition, 14:41–104.

Levy, R. and Jaeger, T. F. (2007). Speakers optimize information density through syntactic reduction. In Schölkopf, B., Platt, J., and Hoffman, T., editors, Advances in Neural Information Processing Systems 19. MIT Press, Cambridge, MA.

Lounsbury, F. G. (1954). Psycholinguistics: a survey of theory and research problems. The Journal of Abnormal and Social Psychology, 49(4, Pt.2):93 – 125.

Maclay, H. and Osgood, C. E. (1959). Hesitation phenomena in spontaneous english speech. Word-Journal of the International Linguistic Association, 15(1):19–44.

Madigan, S. A. and McCabe, L. (1971). Perfect recall and total forgetting: A problem

for models of short-term memory. Journal of Verbal Learning and Verbal Behavior, 10(1):101–106.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics, 19(2):313–330.

Meringer, R. and Mayer, C. (1895). Versprechen und Verlesen: Eine Psychologisch-linguistische Studie. G.J. Göschen, Stuttgart.

Miller, T. (2009). Word buffering models for improved speech repair parsing. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2, EMNLP '09, pages 737–745, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miller, T. (2010). Generative Models of Disfluency. PhD thesis, University of Minnesota.

Oviatt, S. (1995). Predicting spoken disfluencies during human–computer interaction. Computer Speech & Language, 9(1):19–35.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06).

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In Proceedings of NAACL HLT 2007, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Roark, B. (2001). Probabilistic top-down parsing and language modeling. Computational Linguistics, 27(2):249–276.

Schuler, W., AbdelRahman, S., Miller, T., and Schwartz, L. (2010). Broad-coverage incremental parsing using human-like memory constraints. Computational Linguistics, 36(1):1–30.

Shriberg, E. (1994). Preliminaries to a Theory of Speech Disfluencies. PhD thesis, University of California at Berkeley.

van Schijndel, M., Exley, A., and Schuler, W. (2012). Connectionist-inspired incremental pcfg parsing. In Proceedings of the 3rd Workshop on Cognitive Modeling and Computational Linguistics, pages 51–60. Association for Computational Linguistics.

van Schijndel, M., Exley, A., and Schuler, W. (2013). A model of language processing as hierarchic sequential prediction. Topics in cognitive science, 5(3):522–540.

Zwarts, S., Johnson, M., and Dale, R. (2010). Detecting speech repairs incrementally using a noisy channel approach. In Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1371–1378, Beijing, China. Coling 2010 Organizing Committee.