

Segmentation and Feature Selection for Conversational Speech
Syntactic Language Models

William Patrick McNeill

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Arts

University of Washington

2006

Program Authorized to Offer Degree: Linguistics

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

William Patrick McNeill

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Mari Ostendorf

Emily M. Bender

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Segmentation and Feature Selection for Conversational Speech
Syntactic Language Models

William Patrick McNeill

Chair of the Supervisory Committee:
Professor Mari Ostendorf
Electrical Engineering & Linguistics

This thesis investigates the use of syntactic information to improve speech recognition accuracy. Previous work has shown that parser-based language models can be used to improve speech recognition and, separately, that the segmentation of speech into sentence-like units has an impact of parser performance. This thesis brings these findings together by investigating two research questions: 1) can good segmentation improve the accuracy of a speech recognition system that uses syntactic information, and 2) which aspects of the syntactic structure represented in a given segment's parses are most useful for improving recognition accuracy? The system implemented for this thesis resegments word lattices generated by the SRI 5×RT speech recognizer into sentence-like segments which are then converted to N-best lists. The N-best hypotheses are parsed, and syntactic information is extracted from these parses for use in a discriminative model that reranks the N-best lists. Segmentation experiments show significant potential WER gains in the comparison of the oracle and baseline segmentations, although the automatic segmentation schemes investigated do not realize these gains. Experiments with different knowledge sources demonstrate that syntactic information can improve recognizer accuracy. Lexicalized non-local syntactic features that in previous work have proven useful for identifying high-quality parses are also

shown to be useful for recognition accuracy. The best combination of syntactic features achieved a 6% relative reduction in WER on the oracle segmentation and a 3% relative reduction in WER on automatically detected segmentations.

TABLE OF CONTENTS

	Page
List of Figures	iii
List of Tables	iv
Chapter 1: Introduction	1
1.1 Speech Understanding	1
1.2 Syntax, Parsing and Segmentation	1
1.3 A Motivating Example for Syntactic Reranking	4
1.4 Research Questions	8
1.5 Outline	8
Chapter 2: Background	9
2.1 Speech Recognition	9
2.2 Language Modeling	10
2.3 Parsing	13
2.3.1 Probabilistic Grammars	13
2.3.2 Parse Quality Evaluation	15
2.3.3 Discriminative Parse Reranking	16
2.4 Parser-Based Language Models	18
2.5 Segmentation	20
2.5.1 Segmentation in Speech	20
2.5.2 The Effect of Segmentation on n -grams	20
2.5.3 The Effect of Segmentation on Parsing	21
2.5.4 Automatic Segmentation	22
2.6 Summary	24

Chapter 3: Methods	26
3.1 Corpus	26
3.2 System Architecture	27
3.2.1 Speech Recognition	28
3.2.2 Resegmentation	30
3.2.3 <i>N</i> -best List Generation	31
3.2.4 Parsing	31
3.2.5 Feature Extraction	32
3.2.6 Parse Reranking	36
3.3 Evaluation	38
Chapter 4: Experiments	39
4.1 Experimental Variables	39
4.1.1 Segmentation Conditions	39
4.1.2 Reranker Feature Sets	41
4.2 Results	41
4.2.1 Segmentation Results	42
4.2.2 Feature Set Results	46
4.3 Error Analysis	48
4.3.1 Methods	48
4.3.2 Qualitative Observations	50
4.3.3 Segmentation and Accuracy	52
Chapter 5: Conclusions	54
5.1 Contributions	54
5.2 Future Work	55
Bibliography	57
Appendix A: Discriminative Learner Modifications	60
Appendix B: Notation Used in This Thesis	62

LIST OF FIGURES

Figure Number	Page
1.1 A syntactic parse of “The boy in the green hat is elated”	3
1.2 <i>N</i> -best list for “Kim flew the big red airplane”	5
1.3 <i>N</i> -best list for “Kim flew the big red airplane” with parses	6
2.1 A context-free grammar parse of “the boy won a prize”	13
3.1 System architecture	28
3.2 <i>N</i> -best resegmentation using confusion networks	29
3.3 Word and parse hypothesis features	33
4.1 WER obtained by reranking with different feature sets	43
4.2 Example error analysis alignment	50

LIST OF TABLES

Table Number	Page
3.1 Switchboard data partitions	27
3.2 Reranker feature descriptions	32
4.1 Segmentation conditions	40
4.2 Reranker feature combinations	41
4.3 Word error rate results	42
4.4 Significance p -values across segmentations	45
4.5 Significance p -values across feature sets	46
4.6 Aligned and unaligned segment counts	52
4.7 Improved segments	53

ACKNOWLEDGMENTS

I am grateful to my advisors Mari Ostendorf and Emily Bender for their insight and guidance, to Jeremy G. Kahn and Dustin Hillard for their help in implementing this system, and to my wife, Jodi, for her support through it all.

DEDICATION

To Jodi

Chapter 1

INTRODUCTION

1.1 Speech Understanding

In computational linguistics, speech understanding is the task of transforming an audio speech signal into a useful non-audio digital representation. The most basic speech understanding task is speech recognition, the production of a transcript of what was said. This is an interesting challenge in its own right and a precondition for downstream applications such as speech-based extraction and summarization and human-computer dialog systems.

This thesis describes a system that uses syntactic models to improve the accuracy of English conversational speech recognition. Most previous work in stochastic computational syntax has focused on two areas. The first is the extraction of the syntactic information itself, developing techniques for selecting automatic syntactic parses that resemble what would be created by a human annotator. The second is the use of syntactic information to build language models, mathematical models that quantify the relative linguistic coherence of strings of words. This thesis falls into the second category, focusing on improving language models for speech recognition, but leverages recent advances in the the first category, specifically work on parsing spoken language.

1.2 Syntax, Parsing and Segmentation

Consider the following sentence.

(1.1) The boy in the green hat is elated.

Our inherent native-speaker knowledge of the structure of English allows us to make a number of statements about the relationships between the words in (1.1). For example, we know the first “The” attaches a quality of specificity to the boy and not the hat; likewise, “green” tells us something about the hat and not the boy. Though these examples indicate the importance of linear proximity for determining the relationships between words, linear proximity is not the whole of the story, because it is the boy who is elated and not the hat. For the purposes of determining the relationship between these words, the fact that “elated” is the fourth rather than the second or fifth word after “boy” is irrelevant (consider “The boy is elated”). Similarly irrelevant is the fact that “hat” is the first word after “green” (consider “The boy in the green oversized Tyrolean hat”). Theoretical syntax concerns itself in large part with the elucidation of structures which *are* relevant for determining these kinds of relationships. Typically, the relevant structures are pictured as tree diagrams like the one in Figure 1.1. Though it is doubtful that any contemporary theory of syntax would draw a picture that looks exactly like this, Figure 1.1 illustrates the key insights from theoretical syntax that will be used here: sentence structure can be analyzed in terms of trees which are built up out of possibly recursive constituents (e.g. [_{VP} is elated], [_{NP} The boy in the green hat], [_{NP} the green hat]), each of which contains a single syntactically and semantically most prominent word (e.g. “boy” in [_{NP} the boy in the green hat], “is” in [_{VP} is elated]) called its head.

In computational linguistics, these insights inform the design of parsers, programs which take a string of words as input and produce as output a tree structure like the one in Figure 1.1. A wealth of quantitative information can be extracted from these tree structures—for example, we can count the frequencies of various topological relationships—which can be used to put our syntactic insights into a form that allows them to be combined with other mathematical models of natural language. One of the

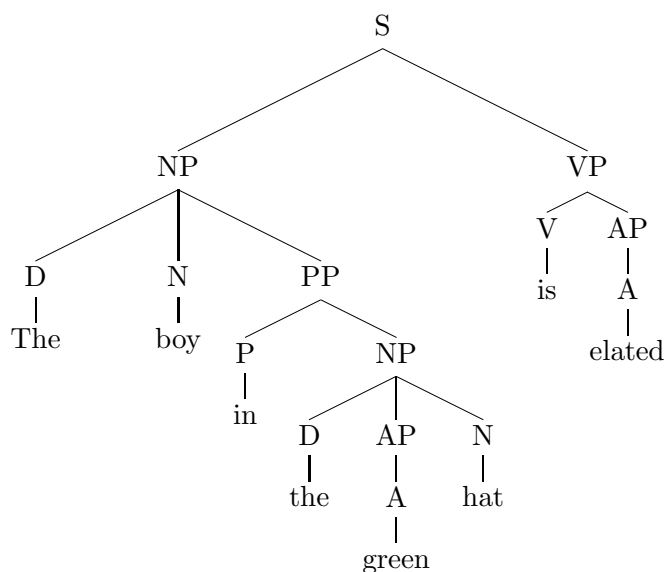


Figure 1.1: A syntactic parse of “The boy in the green hat is elated”

challenges that this work addresses will be that of identifying which aspects of the tree structure it is productive to quantify and how best to go about that quantification.

Underlying all of the preceding discussion has been the notion of a *sentence*. Consider the following string of words.

(1.2) The boy in the green hat is elated. He just won a costume contest.

Most English speakers—linguists and non-linguists alike—would perceive (1.2) as consisting of two distinct entities called sentences, demarcated here by periods and capital letters. Much of theoretical syntax would examine each sentence in (1.2) in isolation from the other. Similarly, automatic parsers would produce two separate trees for “The boy in the green hat is elated” and “He just won a costume contest” with nodes in one bearing no topological relationships to nodes in the other. This is a useful simplification, but it avoids the question of how the separate sentences are identified in the first place. Even in English-language writing, where the notion is fairly well

codified and punctuation serves as a guide, the question can be tricky. (Do independent clauses conjoined by a semicolon count as one sentence or two? What about brief sentence fragments composed in a purposefully telegraphic style for literary effect?) In the speech domain this ambiguity is compounded by many factors including performance errors (called “disfluencies” in the speech understanding literature), the lack of punctuation, and the general tendency for speech to be less likely than writing to conform to prescriptive notions of correct structure. Simply formulating criteria for the demarcation of sentence-like units in speech is a challenging task. In the speech *recognition* domain, lexically-based criteria must be able to work with transcriptions that may contain speech recognition errors. Because of the difficulty of the sentence-identification task and its centrality to the whole notion of parsing, a major focus of this thesis will be on the impact of dividing a word stream into segments amenable to the task of syntactic language modeling.

1.3 A Motivating Example for Syntactic Reranking

Before moving on to the broad research goals of this thesis, I will present a motivating example in this section of how syntactic information can be used to improve speech recognition. This example will introduce both the general problem and the outline of the way in which this thesis tackles it.

For a given audio segment, a speech recognizer produces a ranked list of N word sequence hypotheses called an N -best list. Since speech recognizers are not perfect, these hypotheses contain errors. The recognizer uses acoustic and language models to assign a probability to each word sequence hypothesis, allowing the word sequence hypotheses $\langle s_1 \dots s_N \rangle$ in the list to be sorted in order of recognizer confidence. But since these models are also not perfect, there is no guarantee that the top hypothesis will have the fewest errors. For example, a speech recognizer presented with the spoken sentence “Kim flew the big red airplane” might produce the N -best list shown in Figure 1.2, which gets most of the words correct but erroneously ranks the hypothesis

s_1	Kim to the big red airplane
s_2	Kim flew the big red airplane
s_3	Kim knew the big red airframe
	⋮
s_N	trim sure fair lead game

Figure 1.2: N -best list for “Kim flew the big red airplane”

Kim to the big red airplane as the best guess. Even without hearing the audio, a human being looking at this list could probably guess that its ordering is incorrect, and that a better N -best list would place s_2 before s_1 . An automated system might try to discover the same thing by building an additional language model that reranks the N -best list and then combines the reranked list with the original one in the hopes of pushing better hypotheses towards the top.

To be effective, this second system must incorporate information different than what was available to the recognizer; otherwise, it is likely to simply reproduce the recognizer’s ordering. One way is to model phenomena that the recognizer does not take into account. In this particular case, syntactic phenomena might be helpful for distinguishing between the top two hypotheses, for though the words *to* and *flew* may be acoustically confusable, and the sequence *to the big* no rarer than *flew the big*, the phrase *Kim flew the big red airplane* is a reasonable English sentence while *Kim to the big red airplane* is syntactically odd.

A parser can be used to quantify this syntactic distinction. For each of the word sequence hypotheses s_i in our N -best list we can generate a set of parses $\langle t_{i1} \dots t_{iM_i} \rangle$, as in Figure 1.3. Because s_2 is more syntactically well-formed than s_1 , we would expect the former’s parses to be better than the latter’s. An automatic system that reranked the t_{ij} based on some measure of their syntactic quality could learn to rank at least one of the t_{2j} ones higher than the t_{1j} ones. Taking the word sequence

s_1 Kim to the big red airplane	t_{11}
	t_{12}
	\vdots
	t_{1M_1}
s_2 Kim flew the big red airplane	t_{21}
	t_{22}
	\vdots
	t_{2M_2}
s_3 Kim knew the big red airframe	t_{31}
	t_{32}
	\vdots
	t_{3M_3}
\vdots	
s_N trim sue fair lead game	t_{N1}
	t_{N2}
	\vdots
	t_{NM_N}

Figure 1.3: N -best list for “Kim flew the big red airplane” with parses

hypothesis that generated the best parse would then give us the correct transcription s_2 . It would not matter which of the t_{2j} parses was designated the best one as long as one of them led us back to *Kim flew the big red airplane*. In fact, it might even be helpful to combine the relevant information from all the individual parses for a given word sequence hypothesis into a single score or vector of scores and then choose the best word sequence hypothesis based on these aggregate objects.

Of course, even if the syntactic model can improve on the recognizer rankings, the latter should not be disregarded. For example, *Kim flew the big red airplane* is as syntactically well-formed as *Kim knew the big red airframe*, so domain-dependent language modeling and acoustic information from the recognizer needs to be incorporated in such a way as to prefer parses of the former.

Finally, suppose what the speaker actually said was “Kim flew the big red airplane. It was sunny that day.” If this audio was segmented between *airplane* and *it*, there would be two N -best lists and the parse reranking technique would work as described above. If, however, it was segmented so that one of the N -best lists contained hypotheses like *to the big red airplane it was* and *flew the big red airplane it was*, a parse-based reranker would be of little use because both of these utterances are equally syntactically odd. Because parsers are designed to deal with sentences, the more sentence-like the word sequence hypotheses on which they operate, the more discriminative the parse information will be.

The preceding example illustrates the broad outline of the parse reranking strategy employed in this thesis. The details lie in the particular manner in which desirable segmentations are produced, which aspects of the parses are measured quantitatively, and how this information is combined with the original ranking proposed by the recognizer to produce the best word sequence hypothesis guess.

1.4 *Research Questions*

For this thesis I collaborated in the implementation of machine-learning system that uses automatic segmenters and parsers to improve the accuracy of a large vocabulary speech recognizer using a strategy outlined above.¹ I used this system to run experiments that address the following two questions:

1. Can good segmentation improve the accuracy of a speech recognition system that uses syntactic information?
2. Which aspects of the syntactic structure represented in a given segment's parses are most useful for improving recognition accuracy?

1.5 *Outline*

The remainder of the thesis is organized as follows: Chapter 2 surveys the computational linguistics literature relevant to the topics addressed above. It describes past results that motivate this work, including specific software systems that were incorporated into the one described here. Chapter 3 describes the architecture of the system implemented to explore the questions posed in this thesis. Chapter 4 describes the experiments performed and highlights important aspects of their results. Chapter 5 summarizes the key findings of this thesis and outlines avenues of future work.

¹This system was implemented in collaboration with Dustin L. Hillard and Jeremy G. Kahn. Dustin L. Hillard ran the SRI recognizer and the automatic segmentation software. Jeremy G. Kahn ran the software to generate the parses and extract the parse features. The three of us worked together to build the system's infrastructure, which was based in part on software developed for (Kahn 2005).

Chapter 2

BACKGROUND

This chapter describes previous work that has been done in a variety of speech understanding areas including recognition, parsing, language modeling, and automatic sentence detection and outlines how these various strands are incorporated into the research presented in this thesis.

2.1 *Speech Recognition*

Speech recognizers are computer programs that take audio files of human speech as input and produce transcripts of that speech as output. Because of the high degree of complexity and ambiguity present in natural language, recognizers are implemented as stochastic devices which, given some audio signal A , return the most likely corresponding sequence of words \hat{W} out of the set of all possible word sequences V^* over a vocabulary V :

$$\hat{W} = \operatorname{argmax}_{W \in V^*} p(W|A). \quad (2.1)$$

Bayes' Rule can be used to rewrite the conditioning information as follows:

$$\hat{W} = \operatorname{argmax}_{W \in V^*} p(A|W)p(W). \quad (2.2)$$

The first term of the product in (2.2) is called the acoustic model while the second is called the language model. This form of the equation allows the task to be broken down into separate components focused on acoustics and linguistic coherence, respectively, and this division of labor eases the task of implementation. The work here, for instance, focuses entirely on the information captured by the $p(W)$ term.

State-of-the-art large-vocabulary speech recognizers are much more complicated systems than the simple formulation in (2.2) might indicate. A typical system makes multiple passes over the same data, using the results of one pass to adapt the models used in a subsequent pass (Stolcke et al. 2006). Both the acoustic and language modeling components detect and incorporate information from multiple sources including frequency and amplitude properties of the signal, speaker gender, pause, phone, and lexical information. There are typically multiple intermediary stages in which lattices or N -best lists of hypotheses are rescored. This thesis adopts the general strategy of reranking N -best lists so that its techniques can be easily integrated in speech recognition systems of this type.

The accuracy of a speech recognition system is measured by comparing its output to a human annotator’s transcript of the same audio input. The standard measure of accuracy is the word error rate (WER), which is calculated by finding an edit alignment between the recognizer output and the reference transcripts and dividing the number of recognizer errors by the total number of reference words. In terms of the number of correct words, substitutions, deletions, and insertions identified by the edit alignment, the WER is

$$\text{WER} = \frac{\# \text{ errors}}{\# \text{ reference}} = \frac{\text{sub} + \text{del} + \text{ins}}{\text{corr} + \text{sub} + \text{del}}. \quad (2.3)$$

WER will be used as the evaluation criterium in this work. As is conventional in the speech recognition literature, all WER values will be reported as percentages.

2.2 *Language Modeling*

The language model term $p(W)$ in equation (2.2) is a probability distribution over sequences of possible words. The exact manner in which the probability mass is distributed depends on the structure of the model, though the general strategy is to assign higher probabilities to linguistically coherent sequences and lower probabilities

to ones that look more like gibberish. In this domain, the notion of linguistic coherence is typically understood as a degree of similarity to a typical sample drawn from a training corpus, where the notions of typical and similar are defined at some appropriate level of abstraction. Any number of lexical, syntactic and semantic factors may be brought to bear on the task of constructing $p(W)$. A given sequence of words might be deemed more probable if it contained common lexical collocations, if it was arranged into a syntactically valid sentence, or if it served as a relevant response to a statement that occurred earlier in the same conversation. The kinds of linguistic information that can go into a language model are constrained only by the ingenuity of the model designer and the availability of training data, where the latter tends to be the limiting factor. Because of the inherent richness and ambiguity of natural language, the set of possible word sequences V^* tends to be very large, leaving complicated language models vulnerable to problems of training data scarcity. In practice, language models generally assign probabilities based on the lexical or syntactic features of some relevant local context.

The simplest effective variety of language models are n -grams (Jelinek 1997, Manning and Schütze 2001), which build the probability of a word sequence out of the probability of the individual words where the probability of a given word in a sequence is conditioned on the identity of the preceding n words:

$$p(w_i|w_1 \dots w_{i-1}) = p(w_i|w_{i-n+1} \dots w_{i-1}). \quad (2.4)$$

For reasons of training data scarcity, n is a small number (usually in the range of 2 to 5). A basic estimate of the probability in equation (2.4) can be obtained by simply counting word n -tuples in some training corpus, though in practice these distributions have to be smoothed in order to be useful (Chen and Goodman 1996).

From the point of view of theoretical syntax, n -gram models have a number of obvious shortcomings. They are exclusively concerned with the linear proximity of

words, despite the claims presented in Section 1.2 about the inadequacy of linear proximity for explaining language structure. By limiting the horizon of their conditioning to some small value of n , they forgo chances to capture structural dependencies. Their focus on word counts alone also cannot capture higher-level syntactic or semantic structures. Despite these shortcomings, n -gram language models have proven indispensable for language understanding technology. Their simplicity is a virtue, making them fast and easy to incorporate into a speech recognition system at runtime. More sophisticated language models—including the ones used in this work—usually incorporate an n -gram model as a baseline lexical information source.

Perplexity is the standard measure of language model quality (see Jelinek 1997, Manning and Schütze 2001). Given a language model that assigns a probability $P(W)$ conditioned on past context to a word sequence W , the cross entropy of a test set is given by:

$$H(P) = -\frac{1}{n} \sum_{i=1}^n \log_2 P(w_i | w_1, \dots, w_{i-1}) \quad (2.5)$$

where the collection of sentences in the test set is concatenated to form $w_1 \dots w_n$. The cross entropy measures the degree to which our language model matches the true probability distribution of W given a sufficiently large value of n . In the language modeling literature it is conventional to report a related quantity called the perplexity:

$$\text{Perplexity} = 2^{H(P)} \quad (2.6)$$

A higher quality language model that more accurately represents the distribution of words in a test set will have a lower perplexity for that set. Though perplexity is useful for comparing language models discussed in this chapter, no perplexity measure will be generated for the system used in this thesis since it employs a reranking strategy that does not generate a probability term $P(W)$.

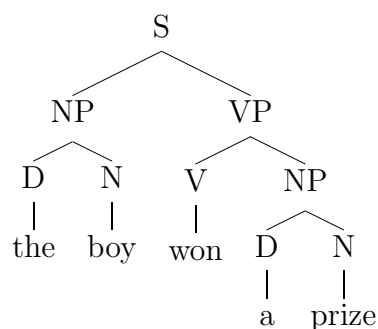


Figure 2.1: A context-free grammar parse of “the boy won a prize”.

2.3 Parsing

2.3.1 Probabilistic Grammars

The basic syntactic concepts discussed in Section 1.2—tree structures, constituency, and recursion—can be formalized using the notion of a context free grammar (CFG) (see Aho and Ullman 1972). A CFG is a 4-tuple of sets $G = (P, \Sigma, R, S)$ where P is a set of non-terminal symbols, Σ is a set of terminal symbols, R is a set of rules, and $S \in P$ is a distinguished start symbol. The rules R are of the form $A \rightarrow B_1 B_2 \dots B_n$ where $A, B_i \in (P \cup \Sigma)$. The grammar G defines a set of pairs of the form $\langle s, t \rangle$ where $s \in \Sigma^*$ is a sentence, and t is the parse of s obtained by starting with S and recursively applying rule expansions to the non-terminals until s is generated. For example, the following grammar

$$\begin{aligned}
 P &= \{S, NP, VP, D, N, V\} \\
 \Sigma &= \{a, boy, prize, the, won\} \\
 R &= \left\{ \begin{array}{l} S \rightarrow NP VP, NP \rightarrow D N, VP \rightarrow V NP, \\ N \rightarrow boy, N \rightarrow prize, V \rightarrow won, D \rightarrow the, D \rightarrow a \end{array} \right\} \\
 S &= S
 \end{aligned}$$

will generate the sentences “the boy won a prize”, “a boy won the prize”, “the boy won the boy” and so on. Figure 2.1 shows the parse of the first of these sentences.

For parse information to be incorporated into a stochastic framework, there needs to be some way of assigning probabilities to parses. History-based models (Black et al. 1992, Collins and Koo 2005) provide a generic formulation of the the joint probability $p(s, t)$ of a sentence-tree pair in terms of a sequence of parser operations $\langle o_1, o_2 \dots o_m \rangle$ carried out to generate t from s . For example, given a canonical leftmost expansion order, the tree in Figure 2.1 could be represented as the following sequence of node expansions:

$$\begin{aligned}
 t = \langle & \text{S, NP VP, D N VP, the N VP, the boy VP,} & (2.7) \\
 & \text{the boy V NP, the boy won NP,} \\
 & \text{the boy won D N, the boy won a N,} \\
 & \text{the boy won a prize} \rangle
 \end{aligned}$$

The parse history of t would then be the sequence of CFG parsing operations $\langle o_1, o_2 \dots o_m \rangle$ applied to generate (2.7). History-based models condition the joint sentence-tree probability on this parse history:

$$p(s, t) = \prod_{i=1}^m p(o_i | \langle o_1, o_2 \dots o_m \rangle). \quad (2.8)$$

History-based models are *generative*; that is, they require the model to specify particular functional forms for the conditional probabilities in (2.8). In practice, it is also necessary for models to define equivalence classes between the possible histories to make the estimation of these probabilities tractable.

A common variety of history-based models are probabilistic context free grammars (PCFG) (Charniak 1993, Manning and Schütze 2001), which define equivalence classes such that the probability of a parser operation is conditioned solely on the identity

of the CFG rule it employs: $p(o_i | \langle o_1, o_2 \dots o_m \rangle) = p(o_i) = p(r_i)$, where $r_i \in R$ is the CFG rule used in parser step o_i and the $p(r_i)$ obey a set of well-formedness conditions. If $C(t, r)$ is the number of times rule r appears in parse t , equation (2.8) becomes

$$p(s, t) = \prod_{r_i \in t} p(r_i)^{C(t, r_i)}. \quad (2.9)$$

Many history-based parsing language models use PCFGs as a base, adding conditioning information extracted from particular parts of the derivation to the rule identity. An important type of PCFG is a lexicalized PCFG, which annotates the rules with the head words of the components they generate. This thesis uses a lexicalized PCFG parser, but transforms its output in such a way as to compensate for some of the shortcomings of history-based models that will be discussed in Section 2.3.3.

2.3.2 Parse Quality Evaluation

As with n -gram models, the job of a parsing language model is to assign higher probabilities to more linguistically plausible parses. The notion of plausibility is defined in terms of similarity to a corpus of human-produced parse trees. With n -grams, degree of similarity is determined at the level of lexical collocations; whereas with a parsed corpus it is understood in terms of tree structure. Two tools for evaluating parse quality in this manner are Parseval and SParseval.

Parseval (Black et al. 1991) is used when the words in the parse tree to be evaluated are identical to those in the reference tree. SParseval (Roark et al. 2006) is an enhancement of Parseval that can be used when the words in the trees are not identical, for example when the words in the tree to be evaluated are taken from a speech recognizer transcript which contains errors. Though each tool generates a few different metrics, this thesis focuses on measures of constituent similarity. Parseval’s constituent measure works by comparing the bracketings in a test and reference tree. SParseval’s constituent measure transforms the trees to be compared into dependency

graphs between heads and the words that modify them and compares sets of head-modifier-relationship triples. Both measures calculate precision and recall for the parse elements they count, so parse quality metrics are typically reported as F-scores.

This thesis is concerned with WER rather than parse quality, so it presents no Parseval or SParseval evaluations. However, it utilizes techniques that are known to increase these measures for the sake of improving the quality of the parses used as language models.

2.3.3 *Discriminative Parse Reranking*

Parses are rich data structures. Even very short sentences may generate trees containing tens of nodes, making for a huge combinatorial space of possible parse features. It is difficult to know *a priori* which kinds of features it is useful to model quantitatively. Are counts of sibling non-terminals helpful, or parent-child relationships, or some more complicated structural configuration? The task of building a probabilistic model is greatly simplified if many different features can be proposed and it is left up to the machine learning algorithm to determine which ones are the most useful for a given task.

Unfortunately, the history-based models described in Section 2.3.1 are not amenable to this kind of automatic feature selection process. In part this is because they are generative—the author of a particular stochastic model cannot simply propose a feature but must also specify the functional form used to incorporate it into equation (2.8). The formulation of parse probabilities in terms of parse histories is also problematic, since there are many varieties of parse information (the average lengths of certain types of syntactic phrases, for example) which might be informative to a downstream system, but are difficult to encode in a sequence of parser operations.

An alternative is to assign parse scores using *discriminative* models, which learn correlations between vectors of features and an objective function without requiring that the particular functional form for the feature probabilities be known. A discrim-

inative model that selects the best parse from a group of candidates, for instance, can use the parse quality metrics described in Section 2.3.2 as its objective function and train on high-dimensional feature n -dimensional real-valued vectors $\vec{\phi}(t) \in \mathbb{R}^n$ consisting of possibly-correlated features extracted from parses. Incorporating a potentially informative feature is as simple as adding it to $\vec{\phi}(t)$. Collins (2004) describes several such models that employ a variety of machine learning techniques, including maximum entropy models, conditional random fields (CRF), and support vector machines (SVM).

A difficulty of this type of parse scoring is that there are no efficient strategies for evaluating all possible parses of a sentence, hence an effective strategy is to combine history-based and discriminative models, using a lexicalized PCFG to generate parses for a set of sentences, then reranking those parses with a discriminative model. The features used by the model include the lexicalized PCFG probability, counts of context free rules, various n -grams of non-terminal expansions, and annotations of the parent and grandparent nodes of various structures. The performance is measured by running Parseval on the top-ranked parse for each sentence. Collins and Koo (2005) used this strategy to achieve an F-measure of 89.75% on *Wall Street Journal* text, a 13% improvement over the PCFG baseline of 88.2%. Using a similar set of features Charniak and Johnson (2005) obtained an F-measure of 91.0% on *Wall Street Journal* text, also a 13% reduction in error. Kahn (2005) extended this technique to a case in which the true words were not known, parsing multiple word sequence hypotheses returned by a speech recognizer on Switchboard data and training a discriminative reranker to find the word-and-parse combination with the best SParseval score. By reranking the 20-best parses for the 20-best word sequence hypotheses he achieved an F-score of 65.5%, a 27% relative improvement over the 49.6% F-score for the 1-best word-and-parse hypotheses.

2.4 Parser-Based Language Models

Probabilistic parsers are of interest in this thesis in part because they can be used as language models for speech recognition. To obtain the parsing language model score for a word sequence W , the sequence is subdivided into a sequence of sentences. Each sentence s is then parsed, producing a set of parses $T(s)$. The parser-based language model score for a given s can be obtained by marginalizing over all the individual parse probabilities:

$$p(s) = \sum_{t \in T(s)} p(t, s). \quad (2.10)$$

The product of (2.10) for all the sentences is the language model score for W . Like n -gram models, parser-based language models are trained so as to assign higher probabilities to more plausible strings of words. The incorporation of syntactic information means that they provide a different kind of information from n -grams, however, one sensitive to higher-level structure and long-distance dependencies.

A number of parser-based language models have been implemented to work with both speech and text, demonstrating improvements in both perplexity and WER over n -gram baselines. Here, I will highlight a few instances of WER improvement on speech, since that is the task this thesis is concerned with. The Structured Language Model (SLM) of Chelba (2000) is a shift-reduce parser that conditions probabilities on preceding headwords. When interpolated with an n -gram model, it reduced WER from 13.7% to 13.5% on the DARPA'93 HUB1 evaluation set, a corpus of read *Wall Street Journal* text. An interpolated SLM also reduced WER on Switchboard data from 41.2% to 41.0%. The top-down PCFG parser implemented by Roark (2001) reduced WER on the DARPA'93 HUB1 evaluation set to 15.1% from a baseline of 16.5% achieved by a trigram language model trained on the same data. Charniak (2001) implemented a top-down PCFG parser that conditions probabilities on the labels and lexical heads of a constituent and its parent. In contrast to both the models

in (Chelba 2000, Roark 2001), the Charniak (2001) model conditions the probability of a given word on the identity of at most two preceding lexical items. The relative reliance on structure over lexical identity makes this model distinctly untrigram-like. This model gets a lower perplexity than the Structured Language Model and Roark's model on *Wall Street Journal* treebank text.

While the details of the parsing algorithms and probability models of the above models vary, all are fundamentally some kind of PCFG. A non-CFG syntactic language model that has been used for speech recognition is the SuperARV model. The SuperARV model calculates the joint probability of a string of words and their corresponding super abstract role values, which are tags containing part of speech, semantic and syntactic information. The SuperARV got better perplexity and WER results than both a baseline trigram and the SLM for a variety of read *Wall Street Journal* corpora (Wang and Harper 2002).

The WER results described above are not directly comparable to any produced in this thesis either because they were run on different corpora or with a different baseline speech recognizer. Nevertheless, they demonstrate the general effectiveness of parser-based language models for speech recognition. The system implemented in this thesis uses the parser in (Charniak 2001), in part because of its effectiveness as a language model but also because of its ability to produce high-quality parses. As will be seen in Chapter 3, parser-based language model scores are only part of the information considered by this system. It also incorporates syntactic feature counts that reflect the structure of the generated parses, so it is essential that the system have high-quality parses to work with.

2.5 Segmentation

2.5.1 Segmentation in Speech

For reasons of computational efficiency, speech recognizers work better with short waveforms and so segment the incoming audio signal before attempting to process it. Recognizer segmentation schemes are implemented using simple acoustic criteria, dividing at speaker turn boundaries and after long pauses. The pause-based acoustic segmentation generally yields units that do not correspond to any notion of sentence (Stolcke 1997).

The task of detecting sentences in conversational speech often comes under the heading of metadata annotation, where metadata is comprised of various kinds of not-explicitly-syntactic structural descriptions of speech, which includes sentence segmentation but also includes the identification of disfluencies and discourse information. This thesis takes its definition of a spoken sentence from (Strassel 2003), which spells out criteria for human annotators to use in annotating various metadata events, including the division of a spoken word stream into sentence-like units (SUs).¹ Prosodic, syntactic and semantic information all inform these criteria for selecting units that generally express a single concept but may often be smaller than the typical linguistic notion of a sentence. For example, an SU might be a single noun phrase answer to a question.

2.5.2 The Effect of Segmentation on n -grams

N -gram models typically divide the W for which they calculate $p(W)$ into segments that are padded on both ends with dummy segment boundary tokens and take the

¹There is a lack of consensus in the metadata community as to whether SU stands for “sentence-like unit”, “syntactic unit”, “semantic unit”, or “slash unit”. This thesis is agnostic on this particular piece of nomenclature, and will generally adopt the more neutral term “segment”, which may refer both to linguistically-motivated SUs and to pause-based segmentations returned by a recognizer.

product of those segment probabilities, and so they can be sensitive to issues of segmentation. In particular, if a model trained on text containing complete sentences is used to rescore speech recognizer output with non-sentence-like pause-based segmentation, the mismatch between training and evaluation data may degrade its performance. Stolcke et al. (1997) reduced Switchboard WER from 46.2% to 45.1% by resegmenting pause-based recognizer N -best lists so that they more closely matched the segmentation of the training data for the n -gram model used to rescore them. Stolcke (1997) was able to get similar results without making reference to the training segmentation at run time by doing Viterbi and forward-backwards decoding on a lattice produced by a speech recognizer using a hidden Markov model (HMM) in which the hidden states were the presence or absence of segment boundaries between each word.

2.5.3 The Effect of Segmentation on Parsing

Sentence segmentation has been shown to have a dramatic impact on parser performance on conversational speech both in circumstances where the words are known and when they are detected by a speech recognizer.

Kahn (2005) resegmented human-transcribed conversational speech using two different kinds of segment boundaries: (1) boundaries detected automatically using lexical and prosodic information by a system similar to the one described in Section 2.5.4, and (2) boundaries detected by a decision tree classifier which simulated pause-based recognizer segmentation. All the segments were then parsed using three different PCFG-based parsers trained on human-segmented parse trees. The parses generated from the automatic segmentation were markedly better according to the Parseval measure than those generated from the pause-based segmentation for all three parsers.

Harper et al. (2005) performed a similar experiment in which there were two dimensions of variation: (1) human-labeled vs. automatically-detected metadata, including SUs, and (2) human-labeled vs. speech recognizer word transcripts. Multiple

parsers produced parses for the resulting segments in each of the four conditions. The quality of these parses was evaluated using SParseval. Harper et al. found a statistically significant interaction between transcript and metadata quality in which better parses were obtained using the human-labeled metadata. Further investigations conducted on reference transcripts indicated that the accuracy of the SU segmentation—as opposed to other kinds of metadata such as the annotation of disfluencies or identification of filler words like *uh* and *um*—was most important for the purposes of parse quality.

2.5.4 Automatic Segmentation

To obtain the benefits of properly segmented speech outlined in Sections 2.5.2 and 2.5.3 in a working system, we need a way of automatically improving on speech recognizer segmentation. This can be framed as a classification problem in which each boundary between words can be labeled as either a segment boundary or not.

There are two standard performance metrics for automatic segmentation performance. The sentence error rate (SER) measures the number of incorrectly hypothesized SU boundaries over the total number of SUs in the reference:

$$\text{SER} = \frac{\text{incorrect SU boundaries}}{\text{reference SU boundaries}}. \quad (2.11)$$

A related metric is the word-level SER, which measures the number of incorrectly hypothesized SU boundaries over the total number of reference words:

$$\text{word level SER} = \frac{\text{incorrect SU boundaries}}{\text{reference words}}. \quad (2.12)$$

Stolcke and Shriberg (1996) implemented a system to automatically segment human-annotated Switchboard data using just lexical information: word identity, part of speech, and turn boundaries. In the training data, segment boundaries were

inserted as tokens; in the test data segment boundary tokens were hypothesized at each word boundary using a hidden-event language model, and the best sequence of boundary tokens was found using the Viterbi algorithm.

Shriberg et al. (2000) implemented a system to detect segment boundaries in Switchboard and Broadcast News audio that uses both lexical and prosodic information. Their system predicts segment boundaries given a sequence of words W and a sequence of prosodic features F , including pause, word, and phone duration, and pitch and energy contours. The lexical and prosodic information was combined using both linear interpolation and an HMM model. On Switchboard data they achieved a SER of 22.2% compared to a baseline of 25.8% produced by hypothesizing no segment boundaries.

Liu et al. (2006) implemented a similar system which given words W and prosodic features F predicts a sequence of metadata events E , including SU boundaries. This system augmented the lexical information with n -gram probabilities, part of speech, and automatically-induced semantic classes, and combined the lexical and prosodic information sources using a variety of machine learning frameworks: HMMs, maximum entropy models, and CRFs. The maximum entropy models and CRFs learned weights on various indicator functions which captured cooccurrences between metadata events and various lexical and/or prosodic features. The HMM models estimated their state transition probabilities from the lexical information using the hidden-event n -gram model from (Stolcke and Shriberg 1996), and their observation probabilities from the prosodic information using bagging decision trees that estimated $p(F_i|E_i)$. During decoding, the forward-backward algorithm was used to determine the highest event posterior probability for each interword boundary:

$$\hat{E}_i = \operatorname{argmax}_{E_i} p(E_i|W, F). \quad (2.13)$$

Segment boundaries were then hypothesized for all SU \hat{E}_i events with probabilities

above a certain threshold, where a threshold of 0.5 is the theoretically best value for minimizing the word-level SER defined in (2.12). All three frameworks achieved improvements in the sentence error rate over a baseline hidden-event n -gram model on both Switchboard and Broadcast News data sets.

As part of their investigation of the effect of segmentation on parse quality, Harper et al. (2005) produced different segmentations of reference speech transcriptions by varying the SU probability threshold in (2.13). They found that a threshold of 0.5 produced segmentation with the lowest SER (as expected), but a threshold of 0.35 (smaller, therefore predicting SU boundaries more often and generating shorter segments) produced the highest SParseval F-score. This work indicates that what constitutes the most desirable segmentation may depend on the downstream application.

2.6 Summary

This chapter discussed a number of speech and language processing technologies that are utilized in this thesis along with key prior work that motivates the questions under investigation here. The underlying system is a conversational speech recognizer which produces multiple word sequence hypotheses for a given segment of audio. In order to choose which hypothesis represents the best guess, the system relies in part on mathematical models of word sequences called language models, which quantify the distinction between more and less linguistically coherent utterances. This thesis utilizes a parser-based language model, which works by assigning probabilities to syntactic parses of a word sequence. Past research has shown parser-based language models to be effective for speech recognition. In addition to the generative method of assigning probabilities to parses, this thesis also uses a method of ranking parses using vectors of counts extracted from the parse trees. Parse reranking methods have proven effective for selecting the best parse from a set of candidates, and the discriminative models they employ allow for a very flexible combination of information from multiple sources.

All language models typically break the word stream down into a sequence of segments, where a better match between the segmentation of the training and test data can lead to better model performance. Since parsers work by assigning structure to sentences, they perform better when given sentence-like segments to parse. The segmentation produced by speech recognizers is generally not sentence-like, so this thesis employs automatic systems that use prosodic and lexical information to divide speech recognizer output into segments more amenable to parsing.

Chapter 3

METHODS

This chapter describes a system that uses parses to improve word error rate in conversational speech recognition. The system combines ideas and tools from Chapter 2 to rerank the word hypotheses emitted by an existing large-vocabulary speech recognizer and serves as an experimental framework in which to measure the impact of different segmentation conditions and syntactic information sources on WER.

3.1 Corpus

These experiments used the Switchboard corpus (Godfrey et al. 1992), a collection of English conversational speech. The Switchboard corpus consists of five-minute telephone conversations between strangers on a randomly-assigned topic. The audio is separated into different channels for each speaker. The data from a single channel is referred to as a “conversation side” or simply a “speaker”.

All the data used in this experiment was taken from a subset of Switchboard conversations for which the Linguistic Data Consortium (LDC) has created syntactic annotation in the form of parse trees. These experiments used the original LDC transcriptions of the Switchboard audio rather than the more recent Mississippi State transcriptions (ISIP 1997) because hand-annotated reference parses only exist for the former.

These parses were preprocessed for use in this system following methods described in (Kahn 2005) which are summarized here. Various aspects of the syntactic annotation irrelevant for this task—for example, punctuation and empty categories—were removed. The parses were also resegmented to match the human-produced SU an-

Table 3.1: Switchboard data partitions

Partition	Sides	Words
Train	1042	654271
Dev	116	76189
Test	128	58494

notation in (Meteer et al. 1995), with some additional rule-based changes performed to make these annotations more closely match the conventions prescribed in (Strassel 2003). In the resegmented trees, constituents spanning SU boundaries were discarded, and multiple trees within a single SU were subsumed beneath a top-level SUGROUP constituent. This resegmentation was necessary because the original LDC parses were made without reference to the audio, so some parse trees are associated with segmentations that are not faithful to the speakers’ intent. Some retokenization of contractions was also necessary when going from parses to N -best lists, since the former treats a word like *can’t* as two tokens, *can* and *n’t*, while the latter treats it as a single token.

The corpus was partitioned into training, development and test sets whose sizes are shown in Table 3.1. Results are reported on the test set. The development set was used during debugging but no results from it are reported here.

3.2 System Architecture

The system is illustrated schematically in Figure 3.1 and involves the following steps:

1. Speech recognition that converts the audio input into word lattices;
2. Resegmentation of the word lattices into linguistically coherent units;
3. Generation of resegmented N -best lists from the segments produced in (2);
4. Parsing of the word sequence hypotheses in the resegmented N -best lists;

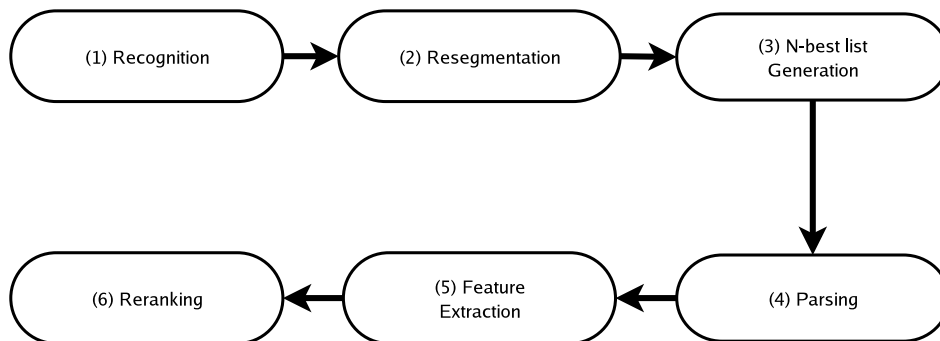


Figure 3.1: System architecture

5. Extraction of recognizer and parser features from the resegmented word sequence hypotheses and parses;
6. Reranking of the N -best lists using a discriminative model.

The following sections describe each of these steps, their inputs and outputs, and the mathematical models and software systems they employ.

3.2.1 *Speech Recognition*

The recognition step (1) takes audio as input and produces word lattices as output. This step uses the the SRI Decipher conversational speech recognition system (Stolcke et al. 2006), a state-of-the-art large-vocabulary speech recognizer that uses various acoustic and language models to perform multiple recognition and adaptation passes. The full system runs in under 20 times real time ($20\times RT$) and has multiple front-ends, each of which produce N -best lists containing up to 2000 word sequence hypotheses per audio segment, which are then combined into a single set of word sequence hypotheses using a confusion network. This system has a WER of 18.6% on the standard NIST RT-04 evaluation test set.

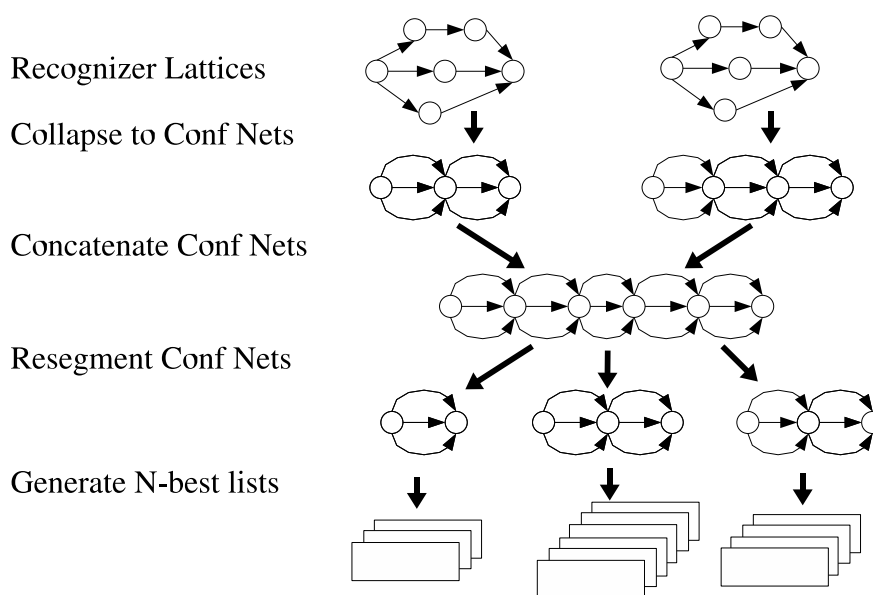


Figure 3.2: N -best resegmentation using confusion networks

Human-annotated reference parses are required for all the data involved in these experiments. This is true even for the test data so as to enable future work in which the quality of the test set parses is examined. Unfortunately, because they are difficult to create, reference parses are in short supply, and all the Switchboard conversations used in the evaluation of this system are already part of the training data for the SRI recognizer. Although it represents only a small part of the training data, there is the danger that this will lead to unrealistically good recognizer performance. This work compensates for this potential danger by using a less powerful version of the full recognizer, the 5 times real time system ($5\times RT$), which has fewer stages of rescoring and adaptation than the $20\times RT$ system. The $5\times RT$ has a WER of 20.2% on the RT-04 test set.

3.2.2 *Resegmentation*

The resegmentation step (2) takes recognizer word lattices as input and produces resegmented confusion networks as output. The system allows for three segmentation types: the default pause-based one returned by the recognizer, automatically-detected segmentations, and an oracle segmentation chosen to match the segmentation of the reference parses as closely as possible.

The resegmentation strategy is depicted in Figure 3.2. First, the lattices from step (1) are converted into confusion networks, a compact version of lattices which consist of a sequence of word slots where each slot contains a list of word sequence hypotheses with associated posterior probabilities (Mangu et al. 2000). Because the slots are linearly ordered, they can be cut and rejoined at any inter-slot boundary. All the confusion networks for a single conversation side are concatenated in chronological order.

For the automatically-detected segmentation conditions, the 1-best word sequence hypothesis for a conversation side is read off the most probable entries in the concatenated confusion networks. Lexical information extracted from these words is then combined with prosodic information extracted from the audio files to produce a segmentation of the 1-best word sequence hypothesis using the HMM variant of the system in (Liu et al. 2006) described in Section 2.5.4. The concatenated confusion network is then cut at locations corresponding to the detected segment boundaries.

The pause-based segmentation is produced by constructing confusion networks directly from the recognizer output without performing any resegmentation. The oracle segmentation is produced by performing a dynamic programming alignment of the concatenated confusion network to the words in the reference parse trees and cutting the confusion network at the parse tree segment boundaries.

3.2.3 *N*-best List Generation

The *N*-best list generation step (3) takes resegmented confusion networks as input and generates *N*-best lists as output. For a given segment, the `lattice-tool` program from the SRI Language Modeling Toolkit (Stolcke 2002) is used to find paths through the confusion network in order of probability, and the *N* most probable paths are emitted as an *N*-best list $\langle s_1 \dots s_N \rangle$, where each s_i is a sequence of words. For the sake of being able to run several different experiments, the *N*-best lists are limited to at most $N = 20$ word sequence hypotheses.

As part of the confusion network generation process, the posterior probability $p_{recog}(s_i)$ of each s_i is also calculated. This is the product of the posterior probabilities of the individual words in the word hypothesis:

$$p_{recog}(s_i) = \prod_{j=1}^{l_i} p_j(w_j|A) \quad (3.1)$$

where $\langle w_1 \dots w_l \rangle$ are the words in s_i . An individual word posterior $p_j(\cdot|A)$ for slot j of the confusion network is estimated from forward-backward lattice acoustic and n -gram language model scores. $p_{recog}(s_i)$ represents the speech recognizer's contribution to the probability model.

3.2.4 *Parsing*

The parsing step (4) takes *N*-best lists as input and produces *N*-best lists annotated with multiple parses as output. The Charniak parser (Charniak 2001) is used to parse the word sequence hypotheses in the *N*-best lists. A parser model is trained on all the reference parses in the training set. For every word sequence hypothesis in a segment s_i for $1 \leq i \leq N$ this model produces parses t_{ij} with corresponding probabilities $p(s_i, t_{ij})$ for $1 \leq j \leq M_i$. To save compute time, at most only the $M_i = 20$ parses are generated.

Table 3.2: Reranker feature descriptions for parse t_{ij} of segment s_i

Feature	Description
$p_{recog}(s_i)$	Recognizer probability
C_i	Word count
B_i	Empty hypothesis flag
$p_{parser}(s_i)$	Parser language model
$p(t_{ij} s_i)$	Parse probability
$p_0(t_{ij} s_i)$	Normalized parse probability
$\vec{\phi}(t_{ij})$	Non-local syntactic features
$E[\vec{\Phi}_i]$	Non-local syntactic feature expectations

3.2.5 Feature Extraction

In the feature extraction step (5), a collection of language modeling features is extracted from the N -best lists generated in step (3) and the parses generated in step (4). The features are listed in Table 3.2. Some are tabulated per parse, so that different parses for the same word sequence hypothesis may have different values. Other features are associated with a particular word sequence hypothesis and their values do not vary from parse to parse. Among the word sequence features, some come directly from the speech recognizer. Others are aggregate values that represent the information present in the set of all the parses for a given s_i . Figure 3.3 shows a schematic of a list of N -best word sequence hypotheses $\langle s_1 \dots s_N \rangle$ with their parses $\langle t_{i1} \dots t_{iM_i} \rangle$ and the features associated with each. All feature values are numeric and all probability scores are in log space.

The reranker features fall into three different categories: recognizer outputs, parse confidences, and non-local parse features. Two recognizer outputs are read directly from the N -best lists produced in step (3) and reflect non-parse information. The first is the recognizer probability $p_{recog}(s_i)$ discussed in Section 3.2.3. As with all n -gram scores, longer sequences will tend to have lower probabilities regardless of their

s_1	$p_{recog}(s_1), C_1, B_1, p_{parser}(s_1), E[\vec{\Phi}_1]$	t_{11}	$p(t_{11} s_1), p_0(t_{11} s_1), \vec{\phi}(t_{11})$
		t_{12}	$p(t_{12} s_1), p_0(t_{12} s_1), \vec{\phi}(t_{12})$
		\vdots	
		t_{1M_1}	$p(t_{1M_1} s_1), p_0(t_{1M_1} s_1), \vec{\phi}(t_{1M_1})$
s_2	$p_{recog}(s_2), C_2, B_2, p_{parser}(s_2), E[\vec{\Phi}_2]$	t_{21}	$p(t_{21} s_2), p_0(t_{21} s_2), \vec{\phi}(t_{21})$
		t_{22}	$p(t_{22} s_2), p_0(t_{22} s_2), \vec{\phi}(t_{22})$
		\vdots	
		t_{2M_2}	$p(t_{2M_2} s_2), p_0(t_{2M_2} s_2), \vec{\phi}(t_{2M_2})$
\vdots			
s_N	$p_{recog}(s_N), C_N, B_N, p_{parser}(s_N), E[\vec{\Phi}_N]$	t_{N1}	$p(t_{N1} s_N), p_0(t_{N1} s_N), \vec{\phi}(t_{N1})$
		t_{N2}	$p(t_{N2} s_N), p_0(t_{N2} s_N), \vec{\phi}(t_{N2})$
		\vdots	
		t_{NM_N}	$p(t_{NM_N} s_N), p_0(t_{NM_N} s_N), \vec{\phi}(t_{NM_N})$

Figure 3.3: Word and parse hypothesis features for the N -best list $\langle s_1 \dots s_N \rangle$

relative linguistic coherence. To prevent the reranker from penalizing sentences for merely being long, we include a second recognizer feature C_i , which is the number of words in the word hypothesis.

There are three parse confidence features, all derived from the the lexicalized-PCFG probability $p(s_i, t_{ij})$ returned by the parser. Since our system is trying to compare parses generated from different word hypotheses, the joint probability is in itself not very useful. If we have $p(s_i, t_{ij}) < p(s_k, t_{kl})$ for some $i \neq k$, it's not clear whether this is because t_{kl} is a worse parse or simply because s_k is a less likely word sequence. All the parse scores used by the reranker are therefore normalized to account for the variability in word hypothesis probability.

The first reranker parse score is the parser language model defined in (2.10), which

is calculated for each s_i by summing the probabilities of all the parses of s_i :

$$p_{\text{parser}}(s_i) = \sum_{k=1}^{M_i} p(s_i, t_{ik}). \quad (3.2)$$

The value from (3.2) is used to calculate a conditional probability of each of the parses given the words:

$$p(t_{ij}|s_i) = \frac{p(s_i, t_{ij})}{p_{\text{parser}}(s_i)}. \quad (3.3)$$

The value from (3.3) is used to calculate the third parse confidence feature, a normalized version of the conditional probability where the most likely parse for a given word sequence hypothesis gets a score of 1:

$$p_0(t_{ij}|s_i) = \frac{p(t_{ij}|s_i)}{\max_{1 \leq k \leq M_i} p(t_{ik}|s_i)}. \quad (3.4)$$

Just as longer sequences will tend to have lower n -gram scores, syntactically complex sequences will tend to produce more parses with lower $p(t_{ij}|s_i)$ probabilities. The features defined by (3.2) and (3.4) are intended in part to prevent the reranker from penalizing sequences for merely being complex just as C_i prevents it from penalizing them for merely being long. We do not know *a priori* which parse score will be more helpful in this regard, so both are included.

Non-local parse features are represented by an n -dimensional vector of integer counts $\vec{\phi}(t_{ij}) \in \mathbb{I}^n$, where \mathbb{I} represents the set of integers, extracted from parse t_{ij} and reflecting various aspects of the parse topology. They are non-local in the sense that they make reference to topology outside the usual context-free conditioning context. The definitions of these features and the software used to extract them are taken from the system in (Charniak and Johnson 2005) described in Section 2.3.3. For example, one element of this vector might count the number of VPs in that parse of length 5 and headed by the word “think”. Because these features are often counts

of the configurations of specific words or non-terminal labels, $\vec{\phi}(t_{ij})$ is a very high-dimensional vector, which was pruned for the sake of computational tractability: for every element in $\vec{\phi}(t_{ij})$ the system keeps a count of all the segments for which that element took on more than one value. Only those features whose values varied between parses in a given audio segment for more than k audio segments were retained for use in the reranking model. This pruning parameter was set to $k = 2000$, which is approximately 2% of total number of training segments for a given segmentation condition and was the value used in parse-reranking experiments in (Kahn 2005). After the pruning is complete, $\vec{\phi}(t_{ij})$ consists of approximately 130,000 elements.

In addition to the integer feature counts there is a per word sequence hypothesis expectation¹ of the non-local features taken over the conditional parse probabilities:

$$E[\vec{\Phi}_i] = \sum_{j=1}^{M_i} p(t_{ij}|s_i)\vec{\phi}(t_{ij}). \quad (3.5)$$

Note that because this an expectation over counts, $E[\vec{\Phi}_i] \in \mathbb{R}^n$. Just as (3.2) generates a per word sequence hypothesis score that is an aggregate of all the individual parse scores, (3.5) is an aggregate of all the individual feature counts for a given s_i . Since for this system, finding the best parse is only a means to the end of finding the best word sequence hypothesis, these aggregate quantities may prove more useful than the values associated with individual parses. It is not clear *a priori* whether this is the case, so we include both kinds of feature in our reranker model.

The ranker must also be able to handle segments that contain only noise, silence, laughter or other non-word events. Empty word sequence hypotheses have $C_i = 0$, $\log p_{\text{parser}}(s_i) = -499$ (an approximation of negative infinity), $p(t_{i1}|s_i) = p_0(t_{i1}|s_i) = 1$, and a $\vec{\phi}(t_{i1})$ containing feature values extracted from the dummy tree [S [-NONE-null]]. To offset the lack of parse information in this circumstance, the reranker

¹Thanks to Jeremy G. Kahn for suggesting this feature.

incorporates a third and final recognizer feature, B_i , the empty hypothesis flag, whose value is defined like so:

$$B_i = \begin{cases} 1 & \text{if } C_i = 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3.6)$$

The $B_i = 1$ flag enables the system to learn the difference between unlikely parses and instances in which there are no words to parse.

3.2.6 Parse Reranking

The reranking step (6) takes feature values for the various word and parse hypotheses as input. The result of training is a discriminative model. During evaluation this model is used to return a reranked order for all the word sequence hypotheses in the N -best lists. If the system is able to discover syntactic information unavailable to the recognizer, the order returned by the reranker will be different than the baseline ordering obtained by ranking according to their $p_{recog}(s_i)$ scores.

During training, the system uses the Charniak parser to generate sets of parses and parse scores for every word sequence and creates a feature vector $\vec{f}_{ij} = \langle f_1 \dots f_m \rangle$ from this data. The elements of \vec{f}_{ij} may be taken from any of the features described in Section 3.2.5, including elements of $\vec{\phi}(t_{ij})$. Different compositions of \vec{f}_{ij} correspond to different sets of syntactic information sources. Each of these reranker feature vectors is paired with an objective function value of $1 - \text{WER}$ for the corresponding s_i , which is calculated by comparing the s_i in the training data with the aligned reference words for that segment.² Note that this objective function value is per word sequence hypothesis, so in any of the feature subsets that contain per parse features there will be multiple parse hypotheses with the same objective function value. This could be the case for any reranking strategy—not just parser-based ones—and does not pose a problem for the learners employed here.

²See Appendix A for details on modifications to the learner that were required to support this objective function.

The \vec{f}_{ij} feature vectors and their corresponding objective values are passed to a discriminative parse reranking learner, which learns the vector of feature weights $\vec{\theta} = \langle \theta_1 \dots \theta_m \rangle$ that yields the smallest value of WER on the training set. The system uses the modeling toolkit from (Charniak and Johnson 2005), but for system speed reasons replaces the maximum entropy learner described in that paper with an average perceptron.

Note that the system needs to run the Charniak parser in order to generate the parses used to train the reranker. The parser used for this purpose cannot be trained on the parses in our reranker training set, since it would then have atypically good performance on that data, giving rise to reranker weights that might not work well on new data. Unfortunately, because of the limited amount of hand-annotated parse tree data, we cannot create a separate training partition just for this model. To work around this problem, the round-robin procedure described in (Collins and Koo 2005) was adopted. Ten models are built, each of which is trained on 9/10 of the conversation sides in the training corpus and used to parse conversations from the remaining tenth. The parses generated from these ten models are what is used to train the reranker. These ten parse models exist solely to generate training parses for the reranker, and are completely separate from the single parse model used to parse the test data in step (4).

During evaluation, the value of the dot product $\vec{f}_{ij} \cdot \vec{\theta}$ is used to rerank the word-parse hypotheses (or just the word sequence hypotheses for feature sets that contain no per parse features). Because of memory limitations, the system was unable to create a single model containing all the training data, so the data was partitioned and 15 separate models were trained. (These partitions are unrelated to the 10-fold parser training partitions.) The ranks produced by these models were then averaged together. The words in the top-ranked word-parse hypotheses were taken to be the system’s hypothesis for a given audio segment.

3.3 Evaluation

This system’s performance is evaluated using WERs generated by the NIST `sclite` scoring tool (NIST 2005) with the words in the reference parses taken as reference. Because we want to compare performance across different segmentations, WERs are calculated on a per-conversation side basis, concatenating all the top-ranked word sequence hypotheses in a given conversation side together in chronological order. The total WER for the system is an average of the WERs for the individual conversation sides weighted by the number of words in a side. When comparing the statistical significance of different results between configurations, the Wilcoxon Signed Rank test provided by `sclite` is used.

Chapter 4

EXPERIMENTS

Chapter 3 described a system that enables the measurement of the impact of different segmentation schemes and syntactic information sources on speech recognizer performance. This chapter describes experiments I performed by running particular configurations of this system.

4.1 Experimental Variables

There are two main experimental dimensions: segmentation and feature sets. I trained models and evaluated a test set over all segmentation/feature set combinations under investigation.

4.1.1 Segmentation Conditions

By changing the configuration of the resegmentation step described in Section 3.2.2, I ran four different segmentation conditions:

- *Pause* ... In this condition I use the pause-based segmentation returned by the recognizer. It is the baseline segmentation condition.
- *Auto₁* ... In this condition I automatically detect segment boundaries using a threshold of $p_{\text{seg}} = 0.5$.
- *Auto₂* ... In this condition I automatically detect segment boundaries using a threshold of $p_{\text{seg}} = 0.35$.

Table 4.1: Segmentation conditions. The average segment length reported here is the number of reference test words from Table 3.1 divided by the number of test segments.

Condition	SU Threshold	Segments		Average Length
		Train	Test	
Pause	NA	54943	5693	10.3
Auto ₁	0.5	86681	8417	6.9
Auto ₂	0.35	96627	9369	6.2
Reference	NA	91254	8779	6.7

- *Reference* ... In this condition I resegment the recognizer output to match the reference segmentations as closely as possible. It is the oracle segmentation condition.

Table 4.1 summarizes the segmentation conditions, including the number of segments and the average segment length in words for each.

The segments in the two automatic conditions were obtained using the SU detection system from (Liu et al. 2006) described in Section 3.2.2. The only difference between them is the value of the threshold parameter p_{seg} used to determine whether a detected SU metadata event with a probability given by equation (2.13) should count as a segmentation boundary. This is a tunable parameter that could take on any value in the range $0 \leq p \leq 1$, but in this work I chose to focus on the values $p_{\text{seg}} = 0.5$ and $p_{\text{seg}} = 0.35$ because the results from (Harper et al. 2005) indicate these are good operating points for maximizing SU prediction accuracy and parse quality, respectively. Note, however, that these are only qualitative guideposts. In particular, the threshold value of 0.35 used in the Auto₂ condition was empirically found to optimize quality on the reference words, not the recognizer transcripts we have here. I did not measure the SU or parse quality created by our system’s segmentation schemes, and additional threshold values could be tried.

Table 4.2: Reranker feature combinations. Additionally all feature sets contain the recognizer features $p_{recog}(s_i)$, C_i and B_i .

Feature Set	Additional Features	Per
Parse	$p(t_{ij} s_i), p_0(t_{ij} s_i)$	Parse
Parse+ParseLM	$p(t_{ij} s_i), p_0(t_{ij} s_i), p_{parser}(s_i)$	Parse
ParseLM	$p_{parser}(s_i)$	Word sequence
ParseLM+Feat	$p_{parser}(s_i), \vec{\phi}(t_{ij})$	Parse
ParseLM+Parse+Feat	$p_{parser}(s_i), p(t_{ij} s_i), p_0(t_{ij} s_i), \vec{\phi}(t_{ij})$	Parse
ParseLM+E[Feat]	$p_{parser}(s_i), E[\vec{\Phi}_i]$	Word sequence

4.1.2 Reranker Feature Sets

For each segmentation condition I ran experiments using a number of feature sets, which are different combinations of the extracted features described in Section 3.2.5. Table 4.2 shows all the feature combinations investigated. Because I am only interested in treating syntactic features as independent variables, all the feature sets also included the recognizer features $p_{recog}(s_i)$, C_i and B_i . For example, the ParseLM+Feat feature set contained the following features: $p_{recog}(s_i)$, C_i , B_i , $p_{parser}(s_i)$, and $\vec{\phi}(t_{ij})$. If any of the features were per parse, the learner reranked parse/word sequence hypotheses and derived the word sequence hypothesis rankings from them. If all the features in the set were per word sequence (instead of per parse), the reranker reranked the word sequence hypotheses directly. (There is one exception for the ParseLM feature set described in Section 4.2.2.)

4.2 Results

Using the system in Chapter 3, I built models and used them to rerank word-and-parse hypotheses for the matrix of experimental variables described in Section 4.1. Table 4.3 shows the WER results of all the segmentation conditions and feature sets, including an oracle value which was generated by selecting the word sequence hypothesis with

Table 4.3: Word error rate results for different sentence segmentations and feature sets. The baseline WER for all segmentations is 22.9%.

Features	Segmentation			
	Pause	Auto ₁	Auto ₂	Reference
Parse	22.8	22.8	22.8	22.8
ParseLM	22.6	22.6	22.8	22.4
ParseLM+Parse	22.6	22.6	22.8	22.4
ParseLM+Feat	22.4	22.4	22.4	21.6
ParseLM+Parse+Feat	22.5	22.4	22.3	21.6
ParseLM+E[Feat]	22.3	22.3	22.3	21.5
Oracle	17.4	16.4	16.0	15.9

the lowest WER for each segment and represents the best the system can do for a given setting of N . Figure 4.1 summarizes these results in a bar chart.

The feature sets are compared against a baseline WER of 22.9% for all segmentations generated by taking the 1-best word sequence hypotheses returned by the recognizer without doing any reranking. This WER is in the same general range as the 20.2% WER returned by the $5\times$ RT system, though the numbers are not directly comparable because they are generated for different test sets.

4.2.1 Segmentation Results

The columns in Table 4.3 list the various segmentation conditions. They are roughly arranged in order of worst to best segmentation, where the Reference is expected to be better than the Pause with the automatic segmentations coming in between, though it is not clear *a priori* which if either of the automatic segmentations should be better than the other.

The oracle numbers from Table 4.3 show a definite improvement in WER as the segmentation improves. The oracle numbers for the different segmentations are all significantly different from each other except for Auto₂ and Reference. In particular,

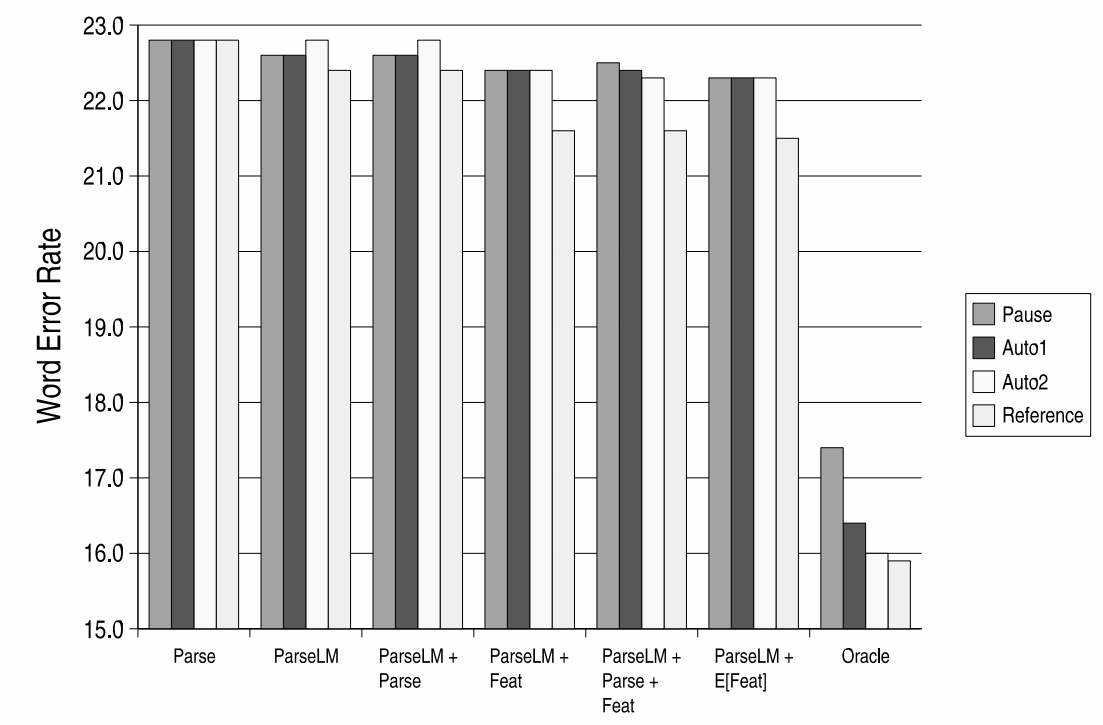


Figure 4.1: WER obtained by reranking with different feature sets

the Pause oracle WER is much larger than the others. This may be due to the fact that the better segmentations tend to be shorter, allowing for richer N -best lists.

Table 4.4 summarizes the statistical significance of the segmentation differences, with the rows containing pairwise comparisons between different conditions and the columns containing different feature sets. If a particular segmentation/feature set combination is statistically indistinguishable, the table contains the word *same*. If the combination is distinguishable, the p -level of significance is indicated.

As expected, the Reference segmentation gives consistently better results than any of the other segmentations. However, neither of the automatic segmentations consistently outperform the pause-based baseline. The Auto_1 segmentation does not perform better than the Pause segmentation for any of the feature sets. There is a significant difference between the Pause and Auto_2 segmentations for the ParseLM and ParseLM+Parse feature sets. For these two feature sets Auto_2 is worse than Pause. Auto_2 also performs slightly better than Auto_1 on the ParseLM+Parse+Feat set. Otherwise, the three non-reference segmentations are statistically indistinguishable. Across segmentation conditions where there is significant difference, the p -value is in a few cases larger for feature sets that do not include syntactic features.

Table 4.4: Significance p -values across segmentations. Statistically indistinguishable pairs are indicated by the word *same*.

Segmentations		Parse	ParseLM	ParseLM+	ParseLM+	ParseLM+	ParseLM+	Oracle
				Parse	Feat	Parse+Feat	E[Feat]	
Reference	Pause	<i>same</i>	0.01	0.01	0.001	0.001	0.001	0.001
Reference	Auto ₁	<i>same</i>	0.001	0.001	0.001	0.001	0.001	0.001
Reference	Auto ₂	<i>same</i>	0.001	0.001	0.001	0.001	0.001	<i>same</i>
Pause	Auto ₁	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>	0.001
Pause	Auto ₂	<i>same</i>	0.05	0.01	<i>same</i>	<i>same</i>	<i>same</i>	0.001
Auto ₁	Auto ₂	<i>same</i>	0.05	0.01	<i>same</i>	0.05	<i>same</i>	0.001

Table 4.5: Significance p -values across feature sets. Statistically indistinguishable pairs are indicated by the word *same*.

Feature Set Pairs			Segmentation			
			Pause	Auto ₁	Auto ₂	Reference
(a)	Baseline	Parse	0.05	0.01	0.001	0.01
(b)	Baseline	ParseLM	0.01	0.01	<i>same</i>	0.001
(c)	Baseline	ParseLM+Parse	0.01	0.01	<i>same</i>	0.001
(d)	Baseline	ParseLM+Feat	0.001	0.001	0.001	0.001
(e)	Baseline	ParseLM+Parse+Feat	0.001	0.001	0.001	0.001
(f)	Baseline	ParseLM+E[Feat]	0.001	0.001	0.001	0.001
(g)	ParseLM	ParseLM+Parse	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>
(h)	ParseLM+Feat	ParseLM+Parse+Feat	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>
(i)	Parse	ParseLM	<i>same</i>	<i>same</i>	<i>same</i>	0.001
(j)	ParseLM+Feat	ParseLM+E[Feat]	<i>same</i>	0.05	0.05	<i>same</i>
(k)	ParseLM	ParseLM+Feat	0.001	0.001	0.001	0.001
(l)	ParseLM+Parse	ParseLM+Parse+Feat	0.05	0.01	0.001	0.001
(m)	ParseLM	ParseLM (per parse)	<i>same</i>	<i>same</i>	<i>same</i>	<i>same</i>

To summarize, improving the segmentation has the potential to help WER performance, particularly when syntactic feature set information is included, but neither of our automatic segmentation schemes is able to realize these gains. Of the two automatic systems, Auto₂ shows the most promise because it has a lower oracle WER and performs better than Auto₁ on one feature set.

4.2.2 Feature Set Results

The rows in Table 4.3 list the results for the various syntactic feature sets in order from worst to best performance. Many of the absolute differences are small, so it is necessary to use significance tests to see which features are actually having an effect. Table 4.5 lists the significant differences between pairs of feature sets for all the segmentations. Again, when a particular feature set pair is statistically indistinguishable, the word *same* appears in the table. Otherwise the p -value level is indicated.

From the results for different groupings of rows in Table 4.3, various conclusions can be drawn. First, rows (a)-(f) indicate that including parse information at all improves over the baseline performance for all the segmentations. The only exception is for the Auto₂ feature sets that contained ParseLM but no syntactic feature information (lines (b) and (c)), where the reranking system does not outperform the baseline. The significance level is markedly improved when syntactic features are included. The parse confidence results in lines (a)-(c)—in particular the parsing language model results on line (b)—are not surprising since previous work of the sort described in Section 2.4 has indicated that parsing language models can improve WER. The results for the feature sets containing Feat, however, indicate that parse reranking techniques that have previously only been used to improve parse quality can also be useful for WER.

The feature set pairs in rows (g)-(h) differ only in whether or not they include the normalized parse confidences in Parse. In neither case does adding the Parse feature set improve the performance. The parse confidences are not useless—they do improve performance in line (a) where they are the only syntactic information utilized—but they do not appear to contain any information that is not reflected in the ParseLM feature set that contains their aggregations.

Lines (i)-(j) examine the effect of various aggregation techniques. Line (i) compares the per-parse information in Parse with a per-word-sequence aggregate of this information in ParseLM and finds the two information sources are only distinguishable in the reference segmentation. Line (j) compares the effect of aggregating parse feature counts by taking their expectation and sees improvements for the two automatic systems, indicating that expectations of feature counts may be a way to partially compensate for less than ideal segmentation.

The feature set pairs in rows (k)-(l) differ only in whether or not they include the syntactic feature information in Feat. In both cases adding the syntactic feature information helped. In particular, compare line (k) to line (g). Both feature

set pairs involve the addition of a kind of per parse information, but the per parse confidences in line (g) have no effect while the per parse feature counts in line (k) improve performance. This is further confirmation that syntactic feature counts are a useful information source for language modeling.

Both items in the pair in line (m) used the ParseLM feature set, which consists of per word sequence values. In one case the system reranked word sequence hypotheses as with all the other ParseLM feature sets in these experiments. In the other case it reranked parse hypotheses with identical feature vectors and objective functions. For all segmentations, the results were statistically indistinguishable. This indicates that, as expected, the reranker model does not suffer performance degradation when presented with multiple identical hypotheses.

To summarize, syntactic information can improve WER. Per parse information is not in general more effective than aggregated per word sequence information quantities like marginalizations and expectations, and adding feature count information improves system performance for all segmentations.

4.3 Error Analysis

The results outlined in the previous section demonstrate that syntactic information and proper segmentation have a positive impact on WER. In this section I examine the reranked word sequence hypotheses produced in these experiments to try and get a sense of where the improvements are coming from. Many different analyses of this output can be performed, but I focus on two here: qualitative observations of improvements in the syntactic quality of the reranked word sequences and an analysis of the interaction between WER and alignment with the reference segmentation.

4.3.1 Methods

In a system like this one, the fundamental error analysis question is for a given N -best list in a test set, what are the error patterns in the top-ranked hypothesis before and

after reranking? To answer this question I used the alignment information generated by the `sclite` tool to do a word-by-word alignment of the baseline 1-best hypotheses, reranked 1-best hypotheses, and references for every conversation side. Figure 4.2 shows an example alignment for a single segment. The top line is the reranked 1-best segment *i go up and you know these things and*. The second line is the 1-best segment before reranking *i go up and owned these things and*. The last line is the reference text *they'd go up you know and do things and*. The vertical bars in this line indicate segment breaks in the reference. Note that the hypothesized and reference segments do not in general align.

For every word I determined whether it was recognized correctly in both the original and reranked N -best lists, and used this information to define the following word error function:

$$Y(w) = \begin{cases} +1 & \text{corrected error} \\ 0 & \text{no change} \\ -1 & \text{introduced error.} \end{cases} \quad (4.1)$$

In Figure 4.2, the plus signs in the third line indicate words for which $Y(w) = +1$, while the minus signs indicate words for which $Y(w) = -1$. The total error change for a segment s is given by

$$\delta(s) = \sum_{w \in s} Y(w). \quad (4.2)$$

Reranking helps the WER for s if $\delta(s) > 0$, hurts if $\delta(s) < 0$, and has no net effect if $\delta(s) = 0$. For the purposes of error analysis it is the change in errors rather than the absolute error count that matters. In Figure 4.2, for example, both 1-best hypotheses contain a substitution error of *i* for *they'd*, so these errors do not count towards δ . For this segment, $\delta = +1$, so reranking helped its performance.

I generated error alignments for the ParseLM+E[Feat] feature sets for all the segmentation conditions. I focused on this feature set because it gave the best performance and so should show the most pronounced effect of syntactic reranking.

Figure 4.2: Example error analysis alignment

EXPT:	i	go up and you know		these things	and
BASE:	i	go up		and owned these things	and
COMP:			- + + - +		
REFR:	they'd go up	you know and		do things and	

4.3.2 Qualitative Observations

Approximately twenty percent of the segments in a given segmentation condition have their WER affected by reranking (that is, have $\delta \neq 0$). Reranking generally only changes a few words in a given segment, and for many of these segments there is no obvious reason as to why it applied. However, a perusal of alignments like the one depicted in Figure 4.2 does turn up few consistent kinds of apparently syntactic improvements. In the examples below, the reranked 1-best hypothesis from the test set appears on top in italics. Aligned beneath it is the baseline 1-best hypothesis, and beneath that is the reference text, which I have punctuated to indicate the reference segmentation. Since the recognizer normalizes text to lowercase, the 1-best hypotheses are presented in all lowercase here.

For a number of segments, the reranked version fixes an error in the determiner of a noun phrase. Examples (4.3)-(4.5) show typical instances in which words like *the* and *a* are corrected.

(4.3) *i do volunteer work for **the** american lung association*
 i do volunteer work for american lung association
 'I do volunteer work for the american lung association.'

(4.4) *but um anyway that's **the** family reunion story*
 but um anyway that's family reunion store oh
 '...but um anyway that's the family reunion story.'

- (4.5) *yeah that would be really time a time saver*
 yeah that would be really time **as** time saver
 ‘Yeah. That would be really time uh a time saver.’

Reranking will also correct errors in the head words of major structural components of the segment. Examples (4.6) and (4.7) show the correction of a subject, and (4.8) shows the correction of a main verb.

- (4.6) *i was nowhere aware of that*
oh was nowhere aware of that
 ‘I was not aware aware of that.’

- (4.7) *they really get uh into it*
yeah yeah really get uh into it
 ‘They uh really get uh into it.’

- (4.8) *that is a pretty big change*
 that **as** a pretty big change
 ‘That is a pretty big change.’

Example (4.7) is a good example of the complementary nature of the syntactic knowledge source, since *yeah yeah* is a very common bigram that an *n*-gram model is likely to prefer over *they yeah*.

Head word correction appears to be particularly effective for contractions. This is unsurprising since a single contraction token may contain the head word information for both the main noun and verb phrases of a clause.

- (4.9) *they’re going to school getting degrees making all this money*
there going to school getting degrees making all this money
 ‘They’re going to school getting degrees making all this money.’

- (4.10) *when they’re young*
 when **there** yeah
 ‘When they’re young.’

Table 4.6: Aligned and unaligned segment counts for the different feature conditions. The number in parenthesis is the percentage of the total number of segments.

	Pause	Auto ₁	Auto ₂	Reference
Aligned	1847 (32.5)	4160 (49.6)	4478 (48.1)	8162 (93.2)
Unaligned	3836 (67.5)	4219 (50.4)	4873 (51.9)	592 (6.8)

(4.11) *so it's uh wound up that uh **we're** the old folks now*
*so it's uh wound up that uh **where** the old folks now*
 'So it's uh wound up that uh we're the old folks now.'

Though examples like these are only a sampling of the improved segments, they represent instances in which the syntactic language model is clearly doing what we expect it to do: correcting errors in words that are acoustically confusable but structurally significant.

4.3.3 Segmentation and Accuracy

The results from Section 4.2.1 indicate that good segmentation helps the syntactic model improve WER. To examine this interaction more closely, I divided all the reranked segments into two classes, *aligned* and *unaligned*, where aligned segments are ones whose edges correspond with segment boundaries in the reference. Table 4.6 shows the numbers and percentages of aligned and unaligned segments for each of the segmentation conditions. Because `sclite` ignores non-word tokens, fragments, and filled pauses, segments that contained no word hypotheses do not appear as part of these tallies, making the aligned and unaligned counts sum to values slightly less than what appear in Table 4.1. Also since the alignment of the reference to the 1-best words may be slightly different than the alignment to the confusion networks, there are a small number of unaligned segments in the Reference condition. Nevertheless, as expected the Pause segmentation has the smallest number of aligned segments,

Table 4.7: Percentage of the changed segments that were improvements.

	Pause	Auto ₁	Auto ₂	Reference
Aligned	71.8	62.3	63.2	68.8
Unaligned	55.7	55.8	55.9	32.1
All	57.9	57.8	58.1	65.1

the automatic segmentations have more, and the Reference segmentation is almost completely aligned.

The function $\delta(s)$ can be used to divide any group of segments into ones that reranking improved, harmed, or did not affect. Ignoring the last category, we can calculate what portion of the affected segments were improvements. Table 4.7 shows the percent of improved segments for all the segmentation conditions. The *All* line shows these percentages over the entire test set, while the *Aligned* and *Unaligned* lines show the percentages for the sets of aligned and unaligned segments, respectively. For the reranking to be doing more good than harm on average, these values must be above 50%. The further away from 50% they are, the more reranking is helping. Table 4.7 shows that reranking is helping for all the segment subsets of all the segmentation conditions. (The unaligned segments for the Reference condition represent a small number of mostly very short segments, and so the 32.1% value is not comparable to the others.) Furthermore, reranking helps more on aligned segments than unaligned ones, with the percent improved for the entire test set falling somewhere in between. The higher values for the aligned segments provide motivation for more work to improve automatic segment detection, while the fact that the unaligned segments remain above 50% indicate that parser-based language models are still robust enough to function in less than ideal segmentation conditions.

Chapter 5

CONCLUSIONS

This chapter surveys the key contributions of this thesis and indicates avenues for future work.

5.1 Contributions

In the course of addressing the two research questions from Section 1.4, this thesis has shown that the right segmentation can produce better parsing language models, and better parsing language models can produce improved WERs.

In answer to the first question about the impact of segmentation on the effectiveness of parser-based language modeling, I have demonstrated that segmentation can have a significant impact on the effectiveness of parsing for speech recognition. The performance of various syntactic language models investigated in this thesis all improved when used on the reference segmentation rather than the one returned by the recognizer. Unfortunately, neither of the two automatic segmentation schemes employed here showed a statistically significant improvement, so for a working system, improvements due to segmentation remain an unrealized potential.

In the course of investigating the second question about which aspects of syntactic structure are most useful for language modeling, I demonstrated that parsing models can be useful tools for improving the quality of recognized speech. The fact that a parsing language model feature improves WER confirms previous work of the sort described in Section 2.4. The fact that discriminative models using parse features help indicates that different varieties of syntactic information—specifically, the kind that comes from discriminative, non-history based models—that had previously proven

useful for improving parse quality are also useful for improving WER.

Finally, the system implemented in this work produced statistically significant reductions in WER. For the pause-based and automatic segmentations the best feature set reduced the WER from 22.9% to 22.3%, a 3% relative improvement. For the reference segmentation, the best feature set reduced the WER from 22.9% to 21.5%, a 6% relative improvement. This is a demonstration of the practical viability of these methods.

5.2 Future Work

This thesis points to a few different avenues of future work. Some are improvements to the system itself; others are ways in which this system could be used as a testing ground for other technologies.

The simplest course of action would be to increase the number of word sequence hypotheses N and/or the number of parses per word sequence hypothesis M to see if the trends observed in this thesis are still manifested at larger beam widths. Since parsing is computationally expensive, the relative importance of these two dimensions is of practical consequence. Using a similar system, Kahn (2005) showed that larger values of N were more useful than larger values of M for the purposes of generating high-quality parses. It would be interesting to know if the same relative beam widths applied to a system that optimizes for WER.

Overall system performance may be improved by tuning the performance of the machine learning component used in reranking. There is nothing about average perceptrons that is uniquely well-suited to this task. Other discriminative techniques—maximum entropy and CRFs, for example—could be tried. Performing normalizations on the syntactic elements of the syntactic feature vector $\vec{\phi}(t_{ij})$ —demoding, or variance normalizations, for example—might also make them more amenable to machine learning techniques. Finally, the composition of the $\vec{\phi}(t_{ij})$ vector could receive further scrutiny. Its components were taken from a system designed to improve parse quality

on written text. By adding, removing, or changing its features, it may be possible to create a reranking system better suited for modeling conversational speech.

It would also be interesting to explicitly examine the quality of the parses most helpful for reranking. Do the parses preferred by a reranker trained on a WER objective function have higher SParseval scores as well? Conversely, do the parses preferred by a reranker trained on a SParseval objective function correspond to word sequence hypotheses with low WERs?

Finally, the WERs produced by this system provide an additional metric for measuring the quality of automatic segmentations. It would be interesting to know whether other operating points for the automatic segmentation threshold p_{seg} produced better WER results or, in failing that, if better results could be produced by different automatic segmentation systems.

BIBLIOGRAPHY

- Aho, Alfred V., and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Englewood Cliffs, N.J.: Prentice-Hall, Inc.
- Black, E., S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing syntactic coverage of English grammars. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, 306–311.
- Black, Ezra, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings of the 5th DARPA Speech and Natural Language Workshop*, Harriman, NY.
- Charniak, Eugene. 1993. *Statistical Language Learning*. Cambridge, Massachusetts: The MIT Press.
- Charniak, Eugene. 2001. Immediate-head parsing for language models. In *Proceedings of the Association for Computational Linguistics*, 116–123.
- Charniak, Eugene, and Mark Johnson. 2005. Coarse-to-fine n -best parsing and Max-Ent discriminative reranking. In *Proceedings of the Association for Computational Linguistics*, 173–180, Ann Arbor, Michigan, June.
- Chelba, Ciprian. 2000. *Exploiting syntactic structure for natural language modeling*. PhD thesis, Johns Hopkins University, Maryland.
- Chen, Stanley F., and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In A. Joshi and M. Palmer (Eds.), *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, 310–318, San Francisco. Morgan Kaufmann Publishers.
- Collins, Michael. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll, and G. Satta (Eds.), *New Developments in Parsing Technology*. Kluwer.

- Collins, Michael, and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1):25–69.
- Godfrey, J. J., E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings ICASSP*, Vol. I, 517–520, San Francisco.
- Harper, Mary, Bonnie Dorr, Brian Roark, John Hale, Zak Shafran, Yang Liu, Matt Lease, Matt Snover, Lisa Young, Robin Stewart, and Anna Krasnyanskaya. 2005. Parsing speech and structural event detection. John Hopkins University Summer Workshop Final Report.
- ISIP. 1997. Mississippi State transcriptions of SWITCHBOARD. <http://www.isip.msstate.edu/projects/switchboard/>.
- Jelinek, Frederick. 1997. *Statistical Methods for Speech Recognition*. Cambridge, Massachusetts: The MIT Press.
- Kahn, Jeremy G. 2005. Moving beyond the lexical layer in parsing conversational speech. Master’s thesis, University of Washington.
- Liu, Yang, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, and Mary Harper. 2006. Enriching speech recognition with sentence boundaries and disfluencies. *IEEE Transactions on Speech, Audio, and Language Processing* 14(5).
- Mangu, Lidia, Eric Brill, and Andreas Stolcke. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech and Language* 373–400.
- Manning, Christopher D., and Hinrich Schütze. 2001. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.
- Meteer, Marie, Ann Taylor, Robert MacIntyre, and Rukmini Iyer. 1995. Dysfluency annotation stylebook for the switchboard corpus. Technical report, Linguistic Data Consortium (LDC).
- NIST. 2005. NIST speech recognition scoring toolkit (SCTK). Technical report, NIST, <http://www.nist.gov/speech/tools/>.
- Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics* 27(2):249–276.

- Roark, Brian, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy G. Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, and Lisa Yung. 2006. SParseval: Evaluation metrics for parsing speech. In *Proceedings LREC*.
- Shriberg, Elizabeth, Andreas Stolcke, Dilek Hakkani-Tur, and Gokhan Tur. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication* 127–154.
- Stolcke, A., C. Chelba, D. Engle, V. Jimenez, L. Mangu, H. Printz, E. Ristad, R. Rosenfeld, D. Wu, F. Jelinek, and S. Khudanpur. 1997. Dependency language modeling. Johns Hopkins University, Baltimore. Center for Language and Speech Processing Research Note No. 24.
- Stolcke, Andreas. 1997. Modeling linguistic segment and turn-boundaries for n-best rescoring of spontaneous speech. In *Eurospeech*, Vol. 5, 2779–2782.
- Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Vol. 2, 901–904, Denver, Colorado.
- Stolcke, Andreas, et al. 2006. Recent innovations in speech-to-text transcript at SRI-ICSI-UW. *IEEE Transactions on Speech, Audio, and Language Processing* 14(5).
- Stolcke, Andreas, and Elizabeth Shriberg. 1996. Automatic linguistic segmentation of conversational speech. In *Proceedings of ICSLP*, 1005–1008.
- Strassel, S. 2003. *Simple Metadata Annotation Specification V5.0*. Linguistic Data Consortium.
- Wang, Wen, and Mary P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, Vol. 10, 238–247, Morristown, NJ. Association for Computational Linguistics.

Appendix A

DISCRIMINATIVE LEARNER MODIFICATIONS

The reranker software described in Section 3.2.6 was originally designed to use SParseval as its objective function. It takes as input the number of hypothesized (n_h), correct (n_c), and reference (n_r) edges from the SParseval dependency graph for each parse hypothesis, which it uses to calculate an F-score objective function. In terms of the input values, the precision and recall are as follows:

$$P = \frac{n_c}{n_r} \tag{A.1}$$

$$R = \frac{n_c}{n_h} \tag{A.2}$$

Choosing an equal weighting for precision and recall gives the following expression for the F-score:

$$F = \frac{2PR}{P + R} = \frac{2n_c}{n_h + n_r}. \tag{A.3}$$

The reranker tool performs the calculation in (A.3) internally. In order to use this software in this experiment, I had to give it WER objective values that it could convert into an F-score. This was done by setting the SParseval counts to be functions of WER parameters as follows:

$$n_r = n_h = \text{corr} + \text{sub} + \text{del} \tag{A.4}$$

$$n_c = \text{corr} - \text{ins}. \tag{A.5}$$

Substituting (A.4) and (A.5) into (A.3) yields the following “F-score”:

$$F = \frac{\text{corr} - \text{ins}}{\text{corr} + \text{sub} + \text{del}} \quad (\text{A.6})$$

$$= 1 - \text{WER}. \quad (\text{A.7})$$

where (A.7) follows from the definition of WER in (2.3). Because the WER can exceed 1, the value in (A.7) may be negative for some sentences, and the reranker tool was modified to accept negative F-score values.

Appendix B

NOTATION USED IN THIS THESIS

V	Vocabulary
V^*	Set of possible strings with vocabulary V
A	Audio signal
W	Word sequence
p_{seg}	SU prediction threshold
N	Number of word sequence hypotheses in an audio segment
M_i	Number of parse hypotheses for the i th word sequence hypothesis
s_i	The i th word sequence hypothesis
$\langle s_1 \dots s_N \rangle$	N -best list of word sequence hypotheses
t_{ij}	The j th parse hypothesis for word sequence hypothesis s_i
$\langle t_{i1} \dots t_{iM_i} \rangle$	M -best list of parse hypotheses for word sequence hypothesis s_i
$p_{\text{recog}}(s_i)$	Recognizer probability based on acoustic and n -gram language model scores
C_i	Word count in hypothesis s_i
B_i	Empty hypothesis flag ($B_i = 1$ when $C_i = 0$)
$p(s_i, t_{ij})$	Probability of t_{ij} for s_i returned by the parser
$p_{\text{parser}}(s_i)$	Parser language model score for s_i
$p(t_{ij} s_i)$	Conditional parse probability of t_{ij} given s_i
$p_0(t_{ij} s_i)$	Normalized parse score of t_{ij} given s_i
$\vec{\phi}(t_{ij})$	Non-local syntactic features of t_{ij}
$E[\vec{\Phi}_i]$	Non-local syntactic feature expectations of s_i

\vec{f}_{ij}	Reranker training vector for (s_i, t_{ij})
$\vec{\theta}$	Reranker feature weights vector