STRUCTURAL EVENT DETECTION FOR RICH TRANSCRIPTION OF SPEECH

A Thesis

Submitted to the Faculty

of

Purdue University

by

Yang Liu

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2004

To my parents and my husband.

ACKNOWLEDGMENTS

I started this research of structural event detection at Purdue University and continued this at ICSI where I have been for the past two and a half years. ICSI has provided a wonderful environment for me to enrich my research on speech and language processing.

I gratefully acknowledge my major advisor Mary Harper for both academic and moral support over the past few years. Even while I have been away from campus, she has been in constant touch via email and phone supporting my research. I have benefited from her insightful guidance and discussion, as well as her encouragement. She has given me the intellectual freedom to do research in spoken language processing and has provided lots of advice. She has taught me how to be a researcher through all these years, while ploughing through the many paper drafts she has revised.

I would like to thank Elizabeth Shriberg and Andreas Stolcke for giving me the opportunity to continue my research at ICSI. I thank them for their valuable suggestions and comments when I encountered difficulties in my research. I have learned from them how to look at a problem both from a scientific and engineering point of view. Especially thanks to Elizabeth Shriberg for teaching me about linguistics as well as for providing academic advice over the past two years.

I thank my other Ph.D committee members: Leah Jamieson and Jack Gandour at Purdue University. They have been very generous with their time and supportive of my research topic. I have benefited from discussions with Leah Jamieson about speech processing in my first two years of study at Purdue University.

Many people at ICSI also deserve acknowledgment. On the academic front, Barbara Peskin shared her vision of the entire structural event detection project, and at the same time was always willing to spend her time working out details. Nelson

Morgan as the director of ICSI, has created an excellent environment that nurtures research and learning. Chuck Wooters and James Fung deserve a special thanks for generating speaker diarization results. Jeremy Ang, Kofi Boakye, Barry Chen, Dave Gelbart, Dan Gillick, Andy Hatch, Yan Huang, Adam Janin, Nikki Mirghafori, and Qifeng Zhu are helpful office mates and neighbors at ICSI. They have made my time at ICSI more enjoyable.

There are so many other people who have contributed to my research. Luciana Ferrer at SRI has helped much with prosodic feature extraction. I thank Mari Ostendorf and Dustin Hillard at University of Washington for their collaboration on the structural event detection work. Wen Wang, who finished her Ph.D from Purdue University and is at SRI now, has been so patient with all my questions regarding language models. I am glad that I had the chance to work together with Lei Chen at Purdue University using a multimodal corpus for sentence boundary detection. Nitesh Chawla at CIBC has been a wonderful source for answers to my machine learning questions. Thanks also to Andrew McCallum at University of Massachusetts and Fernando Pereira at the University of Pennsylvania for their support and advice on the CRF model. I also thank Julia Hirsberg and Yoav Freund at Columbia University for their assistance with the boosting algorithm.

Most of all, I thank my family for their support of my education. I would not be able to reach the end of this journey without consistent support and encouragement from my husband. His belief in me has made this thesis possible. The love from my parents and sister has also supported me during the difficult times in my graduate life.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

ABSTRACT

Liu, Yang. Ph.D., Purdue University, December, 2004. Structural Event Detection for Rich Transcription of Speech. Major Professor: Mary P. Harper.

Although speech recognition technology has significantly improved during the past few decades, current speech recognition systems output only a stream of words without providing other useful structural information that could aid a human reader and downstream language processing modules. This thesis research focuses on the automatic detection of several helpful structural events in speech, including sentence boundaries, type of utterance, filled pauses, discourse markers, and edit disfluencies. The systems evaluated combine prosodic cues and textual information sources in a variety of ways to support automatic detection of these structural events. Experiments were conducted across corpora (conversational speech and broadcast news speech) and with different transcription quality (human transcriptions versus recognition output).

The imbalanced data problem is investigated for training the decision tree prosody model component of our system because structural events are much less frequent than non-events. A variety of sampling approaches and bagging are used to address this imbalance. Significant performance improvements are obtained via bagging. Some of the sampling methods are useful depending on the performance metrics used. Sentence boundary detection and disfluency detection tasks are impacted differently by sampling, bagging, and boosting, suggesting the inherent differences between the two tasks.

A variety of methods for combining knowledge sources are examined: a hidden Markov model (HMM), the maximum entropy (Maxent) model, and the conditional random field (CRF). The Maxent and CRF approaches are discriminatively trained

to model the posterior probabilities and thus correlate with the performance measures. They also support the use of more correlated features and so enable the combination of a variety of textual information sources. The HMM and CRF both model sequence information, unlike the Maxent which explicitly models local information. A model that combines these three approaches is superior to any method alone.

Interactions with other research efforts suggest that the methods developed in this thesis generalize well to other corpora (e.g., a multimodal corpus, a multiparty meeting corpus) and to similar tasks (e.g., a gestural model, dialog act segmentation and classification).

1. INTRODUCTION

1.1 Motivation

Speech recognition technology has improved significantly during the past few decades; for tasks involving read or pre-planned speech, recognition accuracy is often greater than 90%. However, the word-level transcription accuracy for spontaneous conversational speech falls far short of this level, generally lower than 80%. The acoustic properties of spontaneous conversational speech are quite challenging to model due to phenomena such as coarticulation, word fragments, and filled pauses. Additionally, disfluencies and ungrammatical utterances pose serious problems for language models (LMs). These factors combine to affect the performance of speech recognizers on spontaneous speech. The following is an excerpt of a transcription of spontaneous conversational speech. Both the human transcription and the recognition output are shown in the below example. The presence of a word fragment in the example is represented by a '-' after the partial word. Word recognition errors in the recognition output have a strikethrough in them, and the corresponding correct words are shown in bold face inside curly parentheses (corresponding to deletion or substitution errors).

Human Transcription:

> *but uh i'm i i i think that you know i mean we always uh i mean i've i've had a a lot of good experiences with uh with many many people especially where they've had uh extended family and i and an- i i kind of see that that you know perhaps you know we may need to like get close to the family environment and and get down to the values of you know i mean uh it's money seems to be too big of an issue wi- with with with with with what's going on today*

Recognition Output:

> *but ~~um that that~~* **{uh i'm i i}** *i think that you know ~~we~~* **{i mean}** *we always uh i mean i've i've had ~~it there~~* **{a}** *a lot of good experiences with ~~the~~* **{uh}** *with many many people especially ~~with have~~* **{where they've}** *had extended family ~~night~~ and i and* **{an- i}** *i kind of see that that you know perhaps you know we may need to like ~~you're~~* **{get}** *close to the family environment and ~~in~~* **{and}** *get down to the values of you know i mean ~~no and~~* **{uh it's}** *money seems to be too big of an issue ~~we would~~* **{wi- with with with}** *with with ~~really was we would~~ what's going on today*

As can be seen from the recognition output example, current automatic speech recognition (ASR) systems simply output a "stream of words". Structural information (such as the location of punctuation, disfluencies, and speaker turns) is missing, making it difficult for a human to read or for downstream automatic processors to deal with. As shown in the example above, even the human transcription, which contains no word errors, is still hard to read due to the absence of punctuation and the presence of speech disfluencies and filler words.

The transcriptions can be marked with different types of structural information to enhance readability or ease downstream processing. In this thesis, the following types of structural events are considered:

- Sentence boundaries: A sentence ends with './' for a statement, '.../' for an incomplete statement, and '?/' for a question in the marked up transcription examples in this thesis.

- Filler words: These include filled pauses (e.g., *'uh'* and *'um'*) and discourse marker words (such as *'you know'*, *'well'*). The tokens '⟨' and '⟩' are used to mark the extent of these filler words.

- Edit disfluencies: Disfluencies are highly prevalent in conversational speech. In this thesis, the term edit disfluency is used for the "disfluencies"[1] with the following structure (see Chapter 3 for more details):

  (reparandum) * ⟨editing term⟩ correction

  The edited portion of a disfluency (i.e., the reparandum) is marked in examples with parentheses '(' and ')'. For example, in '*a a lot*' in the human transcription shown above, the first *'a'* is the reparandum so it should be marked with parentheses. The interruption point (IP) inside the edit disfluency is marked by '*'. The editing term, which follows the IP and precedes the corrections, is optional. The edit disfluency structure is embedded in utterances and so may be preceded and followed by words that are not part of the edit disfluency.

These types of structural information will be described in more detail in Chapter 3. Below is the annotation of our human transcription example.[2] All the words that interrupt the fluency of speech are shown in bold face in this example. Table 1.1 summarizes the meanings of the symbols used in the annotated transcriptions.

> but ⟨**uh**⟩ (**i'm * i * i think that** ⟨**you know**⟩ ⟨**i mean**⟩ **i've**) * i've had (**a**) * a lot of good experiences (**with**) * ⟨**uh**⟩ with (**many**) * many people especially where they've had ⟨**uh**⟩ extended family ./
>
> (**and i * and**) * **an-** (**i**) * i kind of see (**that**) * that ⟨**you know**⟩ perhaps ⟨**you know**⟩ we may need to ⟨**like**⟩ get close to the family environment (**and**) and get down to the values of ⟨**you know**⟩ ⟨**i mean**⟩ .../
>
> (**uh it's**) * money seems to be too big of an issue (**wi- * with * with * with * with**) * with what's going on today ./

The transcriptions containing this structural information are called '**rich transcriptions**' because they contain much richer information than a simple stream of

[1] These disfluencies are also called "speech repairs" in the literature.

[2] The human transcription is used here to illustrate the the importance of structural information in order to factor out the effect of speech recognition errors.

Table 1.1
Symbols used for the structural events in the example of annotated transcriptions.

Symbol	Meaning
./ or .../	sentence boundaries (complete or incomplete)
⟨ ⟩	filler words
( )	reparandum in an edit disfluency
*	interruption point in an edit disfluency

words. Given this structural information (either human annotated or automatically generated), human transcriptions or recognition output can be "cleaned up" for improved readability. For example, if the disfluencies and fillers are removed from the previous transcription and each sentence is presented with the appropriate punctuation, the cleaned-up transcription would be as follows:

> *But i've had a lot of good experiences with many people especially where they've had extended family. I kind of see that perhaps we may need to get close to the family environment and get down to the value of... Money seems to be too big of an issue with what's going on today.*

Clearly this cleaned-up transcription is more readable, is easier to understand, and is more appropriate for subsequent language processing modules.

There has been a growing interest recently in the study of the impact of structural events. Jones et al. [1] have conducted experiments, showing that cleaned-up transcriptions improve human readability compared to the original transcription. Other recent research has considered whether automatically generated sentence information can play a role in parsing. Gregory et al. [2] have found that using sentence-internal prosodic cues degrades parsing performance; however, the method used for automatically generating sentence-internal annotations was not state-of-the-art. On

the other hand, Kahn et al. [3] have achieved significant error reductions in parsing performance when using sentence boundary information from a state-of-the-art automatic detection system.

1.2 Scope of the Thesis

1.2.1 Structural Event Detection Tasks

Automatic structural event detection is a crucial step for improving the readability of speech recognition output and for making spontaneous speech understanding systems possible. The goal of this thesis is to enrich the recognition output with multiple levels of structural information, including sentence boundaries, filled pause and discourse marker words, and edit disfluencies. We will construct and evaluate algorithms that automatically detect such structural event types.

Note that the problem of sentence boundary detection differs from its analog in text processing, which is sometimes called "sentence splitting" or "sentence boundary detection". The goal of the sentence splitting task is to identify sentence boundaries in written text where punctuation is available; hence, the problem is effectively reduced to deciding which symbols that potentially denote sentence boundaries (i.e., . ! ?) actually do. The sentence splitting problem is not deterministic since these punctuation symbols do not always occur at the end of sentences. For example, in *"I watch C. N. N."*, only the final period denotes the end of a sentence. In the sentence boundary detection task using speech, no punctuation is available, yet the availability of speech provides additional useful information.

We will investigate structural event detection across corpora, on both broadcast news and conversational telephone speech. Broadcast news comprises read speech, formal interviews, man-on-the-street interviews, and some spontaneous speech, although not usually conversational. In contrast, telephone conversational speech is spontaneous, and much of it is quite informal. Broadcast news usually has fewer edit disfluencies than spontaneous conversational speech, and many of these may

be caused by reading errors. Our algorithms will be evaluated on both the human transcriptions and recognition output to investigate the effect of incorrect words in ASR output on system performance.

1.2.2 Our Approach to the Problem

The framework of most current speech recognition systems is to find the most likely word sequence given the speech signal. Because the hidden structure of the utterance (sentence boundaries and disfluencies) does not have an explicit acoustic signal,[3] it is hard to integrate the problem of structural event detection with word recognition in current speech recognition systems. Therefore, we will address this problem by using a post-processing approach that generates the structural information after the recognition results are available. Several knowledge sources will be employed, involving both textual information and prosodic cues to reduce ambiguity inherent in one knowledge source. Figure 1.1 shows a diagram of our approach, the final output of which is a rich transcription or cleaned-up transcription. As the figure shows, prosodic information is obtained from a combination of the speech signal and recognition output, which is used to provide word and phone alignments.

In our investigations, textual information is obtained from the word strings in the transcriptions generated either by a human transcriber or by the ASR system. This type of information is no doubt very important. In many cases, people have no problem inferring appropriate structural events from word transcriptions. Some textual cues are quite useful for automatic identification of structural events, for example, words like "*I*" often start a new sentence, and a repeated or revised word string often signals disfluencies. In addition, the syntactic and semantic information derived from the words provides valuable cues for structural event detection.

[3]There are some implicit prosodic cues at the boundary points, which will be described in Chapter 5.

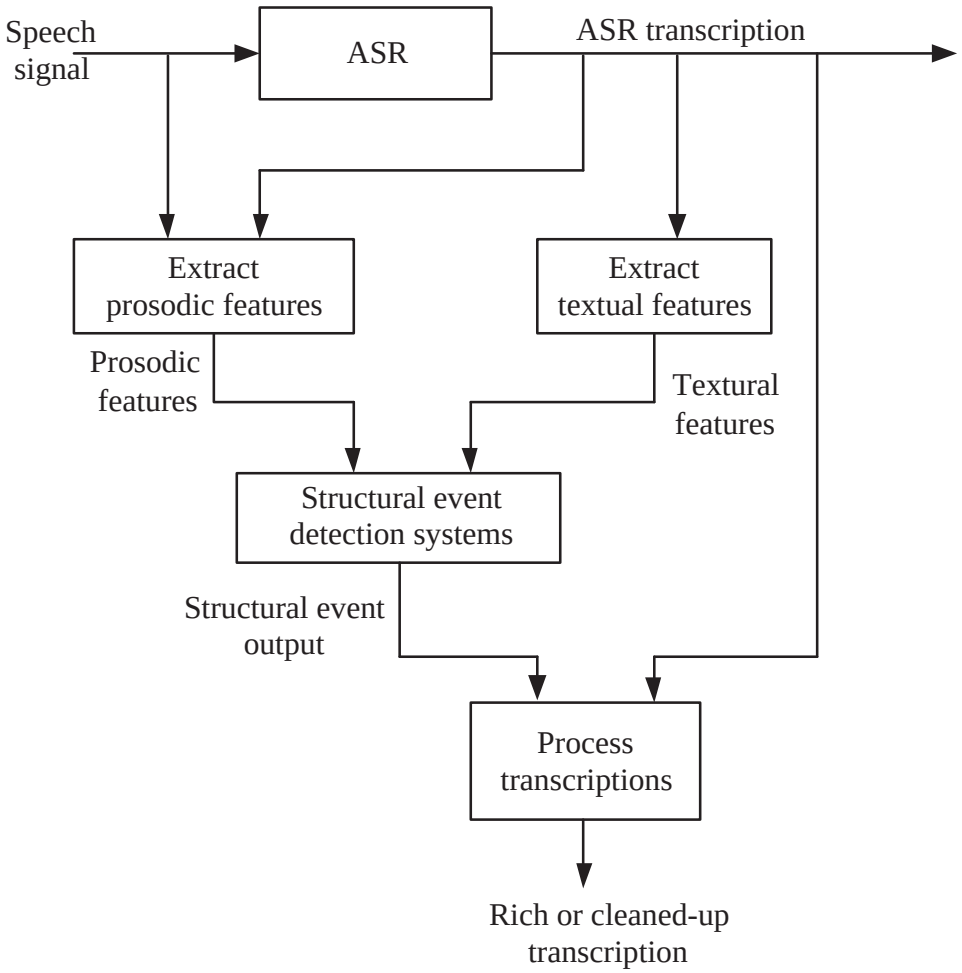

Fig. 1.1. A flow diagram for the automatic structural event detection task.

In some cases, the use of textual information alone may not completely disambiguate structural events. The following example is extracted from the broadcast news data:

Anne what are the chances we'll hear uh something of substance again from the President prior to the vote ?/

And that's a possible next step ?/

A purely textual model would not be able to determine whether the second sentence is a statement or a question. However, the rising tone in the speech signal would enable the listener to determine that a question is intended.

In the face of high word error rates, word level information may be unreliable and possibly misleading. In such a case, the lexical, syntactic, and semantic patterns used for detecting sentence boundaries and disfluencies will be less reliable due to the word errors. The following example compares ASR output with a human transcription of the speech:

ASR output:

It's been a while for the good for the tackle that stuff

Human transcription:

It's been a while since I've uh uh since I've tackled that stuff

It will be difficult, if not impossible, for a word-based language model to identify the repetition or the existing disfluencies using this ASR output.

Prosody, the "rhythm" and "melody" of speech, is important for automating rich transcription. Past research results [4–14] suggest that speakers use prosody to impose structure on both spontaneous and read speech. Examples of such prosodic indicators include pause duration, change in pitch range and amplitude, global pitch declination, melody and boundary tone distribution, vowel duration lengthening, and speaking rate variation. Since these features provide information complementary to the word sequence, they provide an additional potentially valuable source of information for structural event detection. Additionally, since they may be more robust than textual features to word errors, they may provide a more reliable knowledge source.

Textual and prosodic knowledge sources have been exploited in previous research [12, 13, 15–18], and their combination has proven to be beneficial to the performance for structural event detection. This thesis builds upon this prior work that combined these knowledge sources using a hidden Markov model (HMM) approach.

We will focus on developing a richer feature set for these knowledge sources, building more effective models to capture such information, and integrating various knowledge sources for structural event detection by using different modeling approaches.

The investigations in this thesis should help to answer several questions with respect to the automatic detection of structural events: What knowledge sources are helpful? What is the best modeling approach for combining different knowledge sources? How is the model performance affected by various factors such as corpora, transcriptions, and event types?

2. RELATED WORK

In the past decade, a substantial amount of research has been conducted in the areas of detecting intonational and linguistic boundaries in conversational speech, as well as in detecting and correcting speech disfluencies. In this chapter, we introduce research related to the automatic detection of different structural events, namely, sentence boundaries, edit disfluencies, and filler words. For each type, related research is categorized based on what knowledge sources have been used. Additionally, for completeness, studies from linguistics or psychology are discussed where appropriate.

2.1 Sentence Boundary Detection

For speech recognition, "sentences" are usually defined by acoustic segment boundaries that correspond to long stretches of silence or a change of conversational turn.[1] In contrast, linguistic segment boundaries mark a unit that represents a complete idea but may not necessarily represent a grammatical sentence nor begin or end with a long silence or turn change. Experiments by Meteer and Iyer in [19] suggest that language model perplexity can be reduced by working with linguistic segments rather than acoustic segments. Our goal is to automatically find such linguistic sentence-like units.

Some of the previous research has focused on detecting major sentence boundaries;[2] others have investigated detecting subtypes of sentences (e.g., questions, statements). Prior research related to sentence and its subtype detection can be divided

[1]The definition of "turn" varies in the literature. In this thesis, a turn is a portion of speech uttered by a single speaker and bounded by silence from that speaker. See http://secure.ldc.upenn.edu/intranet/Annotation/MDE/guidelines/2004/control_floor.shtml for more details.

[2]The definition of "sentence" varies across these past research efforts. The term used in this thesis will be defined in Chapter 3.

into two categories based on the knowledge sources employed: a text-based approach or an approach using textual and acoustic information. The text-based approach uses only textual information; hence, it is suitable for both transcribed speech and written text. Text-based methods may not be able to resolve some ambiguities using information found in text, as in the example in Section 1.2.2, for which a question type is detected based on the rising tone. A combination approach uses both the acoustic cues and textual information. In most cases, it is difficult to compare the results of prior research since they often differ on the corpora used for training and testing, as well as in the information used by their systems.

2.1.1 Text-based Processing for Sentence Boundary Detection

As mentioned in Chapter 1, the sentence boundary detection problem in written text aims to disambiguate punctuation marks with the goal of identifying sentence boundaries. Palmer and Hirst [20] developed an efficient automatic sentence boundary labeling algorithm, which uses the part-of-speech (POS) probabilities of the tokens surrounding a punctuation mark as input to a feed-forward neural network to obtain the role of the punctuation mark. Because sentence boundaries were not available to their part-of-speech tagger, they used the prior probabilities of all parts of speech for a word. They tested their system on a portion of the Wall Street Journal (WSJ) corpus. Their experiments found that a context of six surrounding tokens and a hidden layer with two units yielded the best accuracy on the test set. When training and testing were conducted using texts in lower-case-only format, the network was able to disambiguate 96.2% of the boundaries. Other approaches have also been used to investigate this problem, for example, Reynar and Ratnaparkhi [21] used a maximum entropy algorithm, and Schmid [22] employed an unsupervised learning method. Walker et al. [23] compared three different methods for sentence boundary detection as a preprocessing step in machine translation. They showed that the maximum entropy method [21] outperforms the other two systems, i.e.,

the direct model and the rule-based system. They also argued that high recall is more important for the application of machine translation: fragmenting sentences is better than combining two sentences. This insight might be useful if we are going to use our structural event detection results in the downstream language processing modules, among which machine translation is one. The sentence boundary problem in text processing is different from that in speech processing in that punctuation information is available in text (although it is not deterministic). However, some knowledge obtained from such a task is useful to our automatic sentence boundary detection in speech, such as the lexical cues that are most effective for determining the role of punctuation.

An automatic punctuation system, called Cyberpunc, which is based only on lexical information, was developed by Beeferman et al. [24]. They counted the occurrence of each punctuation mark in the 42 million tokens of the WSJ corpus and reported that about 10.5% of the tokens in that corpus were punctuation, mostly commas (4.658%) and periods (4.174%). Cyberpunc generates only commas, assuming that sentence boundaries are provided or pre-determined. They extended a language model to account for punctuation by explicitly including commas in an N-gram LM and allowing commas to occur at interword boundaries. Commas were added to the testing word strings by finding the best hypothesis using a Viterbi algorithm. They evaluated this method for generating commas on 2,317 reference sentences of the Penn Treebank WSJ corpus that were stripped of punctuation marks. They obtained a recall rate of 66% and precision of 76% for this comma generation task. The goal of this research differs from sentence boundary detection in speech because the task is to find commas assuming that the major sentence boundaries are known. Beeferman et al. [24] claimed that a punctuation-aware language model can be applied to rescore speech recognition lattices in general, but they did not evaluate this.

Stevenson and Gaizauskas [25] also conducted experiments on identifying sentence boundaries in transcriptions of the WSJ corpus using a memory-based learn-

ing (MBL) algorithm. For each word boundary, they obtained a feature vector of 13 elements from the word and its neighboring words, including the probability of the word starting or ending a sentence, their POS tags, and so on. The precision and recall of their approach was around 35% when case information of the word was removed. The results were much improved when case information was provided to their sentence boundary detection system. Clearly, case information is important for this method, suggesting that it may not extend well to ASR outputs, which do not capture case information and often contain incorrect words.

2.1.2 Combining Textual and Prosodic Information for Sentence Boundary Detection

Some past research has been conducted on combining prosodic information and textual information to find sentence boundaries and their subtypes in speech. It is known that there is a strong correspondence between discourse structure and prosodic information. A comparison between syntactic and prosodic phrasing was presented by Fach [26]. In that study, the syntactic structure was generated by Abney's chunk parser [27] and prosodic structure was given by ToBI label files [28]. This work showed that at least 65% of the syntactic boundaries were prosodic boundaries in read speech.

Chen [29] proposed a method combining speech recognition with punctuation generation based on acoustic and lexical information using a business letter corpus. Punctuation marks were treated as words in the dictionary, with acoustic baseforms of silence, breath, and other non-speech sounds, and her language model was modified to include punctuation. Chen found that 75.6% of all pauses correspond to punctuation marks, and that only 6.5% of the punctuation marks do not correspond to pauses. This finding suggests that pauses are closely related to punctuation in read speech. Chen conducted a speech recognition and automatic punctuation experiment on a business letter with 330 words, read aloud by 3 speakers. For different

testing conditions, Chen reported a result of about 70%-80% accuracy on punctuation placement, but lower accuracy on correct identification of punctuation types. How this result will apply to conversational speech or a larger corpus is unknown.

A sentence boundary recognizer using textual information and pause duration was developed by Gotoh and Renals [15]. In their work, for each interword boundary, a decision is made about whether there is a sentence boundary or not. Their algorithm finds the sequence of sentence boundary classes using speech recognition output by combining probabilities from a language model and a pause duration model. They conducted sentence boundary experiments on 16 hours of Broadcast News corpus using acoustic and duration models trained on 300 hours of acoustic data and using a language model trained on 9 million words. The word error rate (WER) for their test set was 26.3%. They obtained a recall rate of about 62% and precision rate of 80% for sentence boundary detection. Their study found that a pause duration model when used alone performs more accurately than using an N-gram language model for sentence boundary detection. This could be possibly because the language model suffers a lot from the word errors in the recognition output. They found that the result is improved by combining these two information sources.

Shriberg, Stolcke and their colleagues have built a general HMM framework for combining lexical and prosodic cues for tagging speech with various kinds of hidden structural information, including sentence boundaries, disfluencies, topic boundaries, dialogue acts, emotion, and so on [12, 30–33]. Experimental results have shown that the combination of the prosody model and language models generally performs better than using each knowledge source alone.

In [12], Shriberg et al. directly compared two corpora (Switchboard and Broadcast News) on the task of sentence segmentation. Experiments were conducted on both human transcriptions and speech recognition outputs to compare the degradation of the prosody model and LM in the face of ASR errors. They extracted prosodic features such as pause, phone and rhyme duration, and F0 features, as well as other non-prosodic features such as turn change and gender. The features were

used as inputs to a decision tree model, which predicted the appropriate segment boundary type at each interword boundary. They investigated the performance of the prosody model, a statistical LM that captures lexical correlations with segment boundaries, and a combination of the two models. On Broadcast News, the prosodic model alone performed as well as (or even better than) the word-based statistical LM, for both human transcriptions and recognized words. They found that the prosody model often degraded less in the face of recognition errors. Furthermore, for all tasks and corpora, they obtained a significant improvement over the word-only models by combining models. Analysis of the decision trees revealed that the prosody model captures language-independent boundary indicators, such as pre-boundary lengthening, boundary tones, and pitch resets. In addition, feature usage was found to be corpus dependent. While pause features were heavily used in both corpora, they found that duration cues dominated in Switchboard conversational speech; whereas, pitch is a more informative feature in Broadcast News.

Kim and Woodland [16] also combined prosodic and lexical information in a system designed to identify full stops, question marks, and commas in Broadcast News. Their approach is similar to the one used by Shriberg et al. [12]. A prosodic decision tree was tested alone and in combination with a language model, with some improvements reported through the use of the combined model.

Christensen et al. [17] investigated two different approaches to automatically identify punctuation using the Broadcast News corpus. A finite state approach combining a linguistic model with a prosody model significantly reduced the detection error rate and increased the related precision and recall measures, especially when using pause duration. They also showed how prosodic features like pause duration increased detection accuracy for full stops but had very little impact for detecting the other types of punctuation marks. The second approach used a multi-layer perception (MLP) to model the prosodic features. This approach provides insight into the relationship between the individual prosodic features and the various punctua-

tion marks. The results confirmed that pause duration features are the most useful features for finding full stops.

Huang and Zweig [34] developed a maximum entropy based method to add punctuation (period, comma, and question mark) into transcriptions for the Switchboard corpus. Features used in their models involve the neighboring words, the tags (punctuation marks) associated with the previous words, and pause features. They evaluated this approach on both the reference transcription and speech recognition output. Performance was measured using precision, recall, and F-measure. Results showed that performance varies for the different punctuation marks, and adding the bigram type of features (features about the previous and the current position, or the current and the next position) improves F-measure by about 4% over unigram information. They noticed that adding pause information only yields a small gain, in contrast to the results reported for Broadcast news speech (such as [16]). This could be attributed to the different data sets, or to a suboptimal use of pause information in this maximum entropy approach. They observed also that a comma is hard to distinguish from no-punctuation, and that question mark is often confusable with a period. This approach provides a good framework for designing additional features. The maximum entropy approach will be investigated further in Chapter 8.

In the 2003 NIST sentence boundary detection evaluation, all the systems used both prosodic and textual features for sentence boundary detection [35]. The approaches used are similar to the HMM approach used in [12]. For example, one system estimated the likelihood of three classes: complete sentence, incomplete sentence, and non-sentence. They used 48 acoustic-prosodic features estimated for each word boundary, including pause, speaking rate, energy, and pitch features. These prosodic features were used to train a 2-layer neural network. A linguistic subsystem used a trigram LM which has sentence tokens inserted between words. The combined decoder used the likelihood of the sentence classes from the acoustic-prosodic subsystem and the likelihood from the linguistic system, along with a Viterbi algorithm to find the class hypothesis at each word boundary. In another system, a

decision tree was used to predict 4 classes: complete sentence, incomplete sentence, interruption point in edit disfluencies, or non-event boundary. The prosodic features provided to the decision tree are similar to the ones described in [12]. In addition, the posterior probability from the LMs was included as a feature in the decision tree. These two systems were further combined using a 2-layer neural network which uses the minimum square error back-propagation algorithm to hypothesize a binary score at each word boundary. These systems were evaluated on both the Conversational Telephone Speech (CTS) and Broadcast News speech (BN), using both human transcriptions and speech recognition output.

There is also some work that relies on only the prosodic information for finding the sentence units. Wang and Narayanan [36] developed a method that used only the prosodic features (mostly pitch features) in a multi-pass approach. They did not use any word or phone alignment and thus avoid using a speech recognizer. They fit the pitch contour with two linear folds and search for major breaks in the pitch contour. Then in the second pass, sentence boundaries are detected based on some pre-defined rules and statistics. They evaluated this algorithm using a subset of the Switchboard corpus, and obtained a false alarm rate of 17.9% and a miss rate of 7.1%. This result is encouraging since only pitch information is used. However, in conversational speech, pitch may not be a very effective feature for sentence boundary detection. Clearly we would expect that adding additional prosodic and textual information may yield further improvement.

2.1.3 Summary of Past Research on Sentence Boundary Detection

Finding sentence-like units and their subtypes can make transcriptions more readable, while also aiding downstream language processing modules, which typically expect sentence-like segments. Previous work has shown that lexical cues are a valuable knowledge source for determining punctuation roles and detecting sentence boundaries, and that prosody provides additional important information for spo-

ken language processing. Useful prosodic features include pause, word lengthening, and pitch patterns. Past experiments also show that detecting sentence boundaries is relatively easier than reliably determining sentence subtypes or sentence-internal breaks (e.g., commas). The poor performance of sentence-internal structure detection also affects downstream processing, such as parsing [2]. Table 2.1 summarizes important attributes of much of the previous research. Most make use of textual information, either by using a statistical LM or employing other machine learning strategies. The value of adding more syntactic information to the task of sentence detection is an open question. The approaches listed in the first five rows are similar to the approach taken in this thesis, since textual and prosodic information are combined for sentence boundary detection.

2.2 Edit Disfluency Processing

Disfluencies have been investigated using a variety of approaches. Linguists and psychologists have considered disfluencies largely from a production and perception standpoint; whereas, computational linguists have been more concerned with recognizing disfluencies and thus improving machine recognition of spontaneous speech. Although the latter is our main focus, we believe that a better understanding of the underlying theory of disfluency production and its effect on listeners' comprehension can help to construct a better model for the automatic detection of disfluencies; therefore, we will briefly discuss some studies in psychology and linguistics.

2.2.1 Production and Properties of Disfluencies

Disfluency Production

Disfluencies are very common in spontaneous speech. When speakers cannot formulate an entire utterance at once or when they change their minds about what they are saying, they may suspend their speech and introduce a pause or filler before

Table 2.1

A summary of some important prior studies on sentence boundary detection. Column two is the task chosen for each investigation: boundary means the sentence boundary detection task, compared to its subtype or punctuation detection; column three describes the model or the information sources used by each investigation; column four is the corpus on which the experiments were conducted; column five represents whether the experiments were performed on human transcriptions (Ref) or recognition results (ASR). Note that CTS (i.e., conversational telephone speech) is used in the corpus column for those experiments that were conducted on the Switchboard corpus. * Even though no textual information is used in this automatic detection model, "Ref" condition is used in that study for its evaluation.

Investigation	**Classification Task**	**Model**	**Corpus**	**Ref or ASR**
Shriberg et al. [12]	boundary	prosody, word-LM	CTS, BN	Ref, ASR
Gotoh, Renal [15]	boundary	pause, word-LM	BN	ASR
Kim, Woodland [16]	punctuation	prosody, word-LM	BN	Ref
Huang, Zweig [34]	punctuation	Maxent (word, pause)	CTS	Ref, ASR
NIST eval systems [35]	boundary	prosody, word-LM	CTS, BN	Ref, ASR
Beeferman [24]	commas given boundary	word-LM	WSJ	Ref
Stevenson, Gaizauskas [25]	boundary	MBL (word, POS)	WSJ	Ref
Chen [29]	punctuation	punctuation token with acoustic information	a business letter	ASR
Wang, Narayanan [36]	boundary	pitch	CTS	Ref*

continuing, or add, delete, or replace words they have already produced. Spontaneous speech is systematically shaped by the problems speakers encounter while planning an utterance, accessing lexical items, and articulating a speech plan. Speech errors and disfluencies produced by normal speakers have been studied for decades to learn about linguistic production and the cognitive processes of speech planning [37–39].

Disfluency has been used as evidence for cognitive load in speech planning. Oviatt [40] and Shriberg [41] have shown in different types of task-oriented conversations that long utterances have a higher disfluency rate than short ones. This effect may be related to the planning load of the utterance, i.e., speakers have more difficulty planning longer utterances, while making task-oriented plans at the same time. Another observation is that disfluencies occur more frequently at the beginning of an utterance when the utterance is at an early planning stage, providing evidence of the impact of utterance planning on disfluencies.

Clark and Wasow [42] studied the phenomenon of repeated words in spontaneous speech. In their work, repeats are divided into four stages: initial commitment, suspension of speech, hiatus, and restart of the constituent. These four stages correspond to the four components (i.e., reparandum, interruption, editing term, and correction) that have been laid out in Chapter 1 for all edit disfluencies. They proposed a *commit-and-restore model* of repeated words, as well as three hypotheses to account for the repeats, namely, the complexity hypothesis, the continuity hypothesis, and the commitment hypothesis. They hypothesize that the more complex a constituent, the more likely speakers are to suspend it after an initial commitment to it (i.e., complexity hypothesis), and that speakers prefer to produce constituents with a continuous delivery (i.e., continuity hypothesis), and that speakers make a preliminary commitment to constituents, expecting to suspend them afterward (i.e., commitment hypothesis). They analyzed repeated articles and pronouns in two large corpora, the Switchboard corpus and the London-Lund corpus,[3] and found strong empirical evidence to support the proposed commit-and-restore model, along with

[3]See [42] for a description of the corpus.

evidence for all three hypotheses. They noticed that speakers are more likely to make a premature commitment, and then immediately suspend it when the constituent becomes more complex, and that it is more likely that speakers restart a constituent the more that their suspension disrupts the utterance. One example is the frequent occurrence of function words in repeats. It has long been recognized for English that function words are repeated far more often than content words. When speakers want to make an initial commitment to a constituent, the word they mostly commonly use is a function word. Overall, Clark and Wasow [42] found that function words were repeated more than ten times as often as content words, 25.2 versus 2.4 per thousand in the Switchboard corpus. This more frequent occurrence of function words in repeats is explained by the three hypotheses they proposed.

Knowing the types of words that speakers tend to repeat (or revise) is helpful for building a better model of spontaneous speech. For example, when speakers repair a content word, they often return to a major constituent boundary, such as *"on Friday, I mean, on Monday"*. Such an observation is beneficial for defining disfluency patterns and can aid in automatically identifying them.

Effect on Listeners

It is also valuable to understand how human listeners cope with disfluent input. Studies by Lickley [43], Lickley and Bard [5] have shown that listeners generally miss the disfluencies or incorrectly report on the occurrence of disfluencies, suggesting that disfluencies may have been filtered out for utterance comprehension. Psycholinguists believe that disfluencies play specific roles in our communication, such as sending signals to the listener to do things like pay more attention, help the speaker find a word, or be patient while the speaker gathers his or her thoughts. Disfluencies provide information that enables people in a conversation to better coordinate interaction and manage turn-taking [41].

Brennan [44] investigated how comprehension is affected when listeners hear disfluent speech. In her experiments, listeners followed fluent and disfluent instructions for selection of an object in a graphical display. She found that listeners make fewer errors when hearing less misleading information before the interruption points of disfluencies. She also observed that mid-word interruptions are better signals than between-word interruptions that a word was produced in error and that the speaker intends to replace it. This supports Levelt's hypothesis [38] that "by interrupting a word, a speaker signals to the addressee that that word is an error. If a word is completed, the speaker intends the listeners to interpret it as correctly delivered". Brennan also found in her experiments that there is information in disfluencies that partially compensates for any disruption that listeners meet while processing disfluent speech.

Fox Tree [45] studied how naturally occurring speech disfluencies affect listeners' comprehension. She observed that disfluencies do not always have a negative effect on comprehension. For example, repetitions do not hinder the listeners, because they can help listeners to recover information missing in the first occurrence of words that are repeated. However, it does take longer to identify words when there is a false start. When false starts begin utterances, listeners may abort the false starts with no cost to comprehension. But, if false starts are in the middle of utterances, listeners have to figure out where the false start begins, what to abort, and where to attach the restarted information. This process slows down comprehension.

Disfluency Rates

A conservative estimate (excluding silent hesitations) for the rate of disfluencies[4] in spontaneous speech is approximately 6 words per 100 words [45]. There are a variety of factors that may influence disfluency rate.

[4]Speech disfluencies here include fillers.

Disfluency rates vary across different corpora. Oviatt [40] found that people talking on the telephone produced more disfluencies than those talking face-to-face, 8.83 to 5.5 disfluencies per 100 words. Shriberg [41] also found that disfluency rate is lower in speech directed at machines. Differences in disfluency rates in conversations conducted over different media are attributed to the resources these media may offer for coordination. For example, when eye contact and other visual cues such as gesture [46] are available, they can be used to signal such things as the intention to continue speaking or difficulty with an utterance in progress.

Disfluency rate is also affected by the speaker's age, gender, and familiarity with the conversational partners. Shriberg [41] showed that men produced relatively more fillers than women did, but the sexes had similar disfluency rates for the other disfluency types. More fillers may provide a way for men to maintain the floor. Manyhart [47] compared speech produced by children, adults, and the elderly people in Hungarian, and did not observe any differences in the frequency of different types of disfluencies, although children generated more disfluencies per 100 words than the other groups. Males and females differ with respect to the rate of disfluencies, with women generating more.

Bortfeld et al. [48] examined several factors that may affect disfluency rates using a corpus of task-oriented conversations. These factors included speakers' ages, gender, task roles, difficulty of topic domain, and the relationship between speakers. Older speakers produced only slightly higher disfluency rates than young and middle-aged speakers. Overall, disfluency rates were higher when speakers acted as directors or when they discussed abstract figures, confirming that disfluencies are associated with an increased difficulty in planning speech. However, fillers were distributed somewhat differently than repeats or restarts, suggesting that fillers may be a consequence of interpersonal coordination.

2.2.2 Past Research on Automatic Disfluency Detection

Like sentence boundary detection, the research on automatic disfluency detection has been conducted using a text-based approach, or an approach combining textual and prosodic information. It is difficult to compare the results of prior work because the data sets are different and also "disfluencies" mean different things.

Text-based Processing

Bear et al. [49] proposed a two-stage speech repair processing method. The first stage is a simple pattern-matcher, which uses lexical pattern matching rules to retrieve candidate repair utterances. This is done by finding identical sequences of words and pre-specified simple syntactic anomalies, such as "*a the*" or "*to from*". Of the 500 sentences in the ATIS corpus that this algorithm hypothesized as containing a repair, 62% actually did, among which the algorithm made the appropriate correction to 57%. The repair candidates constitute useful input for further processing based on other sources of information such as syntactic and semantic information. In the second step, a natural language processing system is used to distinguish repairs from false positives by either parsing the whole sentence or parsing only localized word sequences identified as potential repairs to avoid the effect due to factors unrelated to the portion with the repair. Bear et al. [49] claimed that acoustic information can be quite effective when combined with other sources of information, noting acoustic differences between the true speech repairs and false positives.

The two-stage approach by Bear et al. [49] promotes the important idea that automatic repair processing might be made more robust by integrating knowledge from multiple sources. The lexical pattern matching approach is computationally tractable and provides reasonable coverage of repair types. However, this method can only detect speech repairs with the predefined patterns. One weakness of this approach lies in the conceptualization of repair types. It is difficult to systematically extend the pattern definition to increase the coverage of such a system because of

the difficulty of listing all the possible repair patterns. In addition, experiments were conducted on the ATIS corpus, which is more template-based than the Switchboard conversational speech, and thus is more amenable to a pattern-based approach.

Charniak and Johnson conducted speech repair detection before parsing Switchboard sentences [50]. A classifier is used to predict the edited words (i.e., the reparandum region of an edit disfluency) based on features such as the POS tags of the preceding word and the following word, and whether or not the word token appears in a "rough" copy, which identifies repeated sequences of words that might be repairs. With the goal of minimizing misclassification, they used a greedy boosting algorithm for classification. On the Switchboard corpus, where 5.9% words are edited words, they obtained an overall misclassification rate of 2.2% with a precision of 94.4% and a recall of 66.8% for the edited words. After detecting and removing the edited words, a statistical parser parses the remaining words in the utterance. The parser achieved a precision rate of 85.3% and a recall of 86.5% on the cleaned-up utterances, but the parsing results were not reported for the original utterance.

Unlike the method used by Bear et al., which uses a parser to help detect repairs, Charniak and Johnson [50] did not feed any information from the parser back to help detect repairs. The edited word detection is based on the assumption that edited words are relatively shallow phenomena that can be detected by using repeated words and POS tags, and that the information provided by a parser is much less critical. However, it is our belief that a variety of knowledge sources are critical for accurate edited word detection, including syntactic information.

In [51], Johnson and Charniak proposed a new noisy channel model for speech repair detection. They used a Tree Adjoining Grammar (TAG) to represent the tree-structured dependencies between the reparandum and correction regions. The source model is a language model (word N-gram or syntactic parser LM), which describes the clean sentence X that does not contain the reparandum. A TAG is used as a channel model that defines the conditional probability of the surface sentence Y given X. Several distributions are estimated: the probability of a disfluency

beginning after a word in X, the probability distribution of the editing term, the probability of a disfluency type given the word in the reparandum and correction region, and the probability that a word M_i in the correction region is a word that is inserted or substituted for a word R_i in the reparandum. The last two distributions are estimated based on the alignment of the reparandum and the correction regions in the training set. Experiments were conducted using the transcriptions of the Switchboard corpus. In testing, all partial words and punctuation are removed from the data to reflect a more realistic testing situation using the speech recognition output (assuming no word errors). A precision rate of 82.0% and recall of 77.8% is obtained when the parser language model is used. This result is significantly better than that obtained by using a word-by-word classifier [50]. The main reason for using the TAG channel model is to model the cross-dependency between the reparandum and correction in a disfluency. These account for the majority of disfluencies in conversational speech; however, these can also be modeled by other simpler methods, without using syntactic parsers. This algorithm does not handle restart disfluencies, which do not have the correction part, or complex disfluencies in which a correction is the reparandum of the following disfluency.

Core and Schubert [52] proposed a framework for handling speech that contains repairs. Assuming that pre-parser repair identification is performed, their parser can use its grammar knowledge and the syntactic structure of the input to correct some errors in the repair identification results. Using the Trains corpus, Core and Schubert obtained a 4.8% increase in the recall rate of speech repairs, but precision dropped from 55.76% to 43.46%. This approach is similar to the experiment by Bear et al. in [49], except that the focus in [49] was on reducing false alarms. Core and Schubert argue that their goal was to increase the recall rate of speech repair detection, which is obtained by trying alternatives with lower parsing probabilities when no parse can be found, and that the "drop in precision is a worthwhile tradeoff as the parser is never forced to accept posited repairs but is merely given the option of pursuing alternatives that include them" [52].

This argument might be acceptable if rich transcriptions (word sequence and the hypothesized disfluency information) are used as input to a language processing module; whereas, if the goal is only to yield cleaned-up transcriptions, then more false alarms (i.e., low precision) will remove fluent utterances and reduce important information. The parser used by Core and Schubert requires the speech repair information as input, so the parser can use its syntactic knowledge to correct speech repair identification errors in the input. However, if the ASR output contains many incorrect words, the parser may not be robust. Further evaluation of this approach is needed.

Zechner [53] used a word-based approach for disfluency and sentence boundary detection in a dialogue summarization system.[5] He used a POS tagger which includes, in addition to the standard Switchboard treebank tag set, tags for the disfluent regions and special purpose words, including co-ordinating conjunctions (e.g., *"and", "then"*), discourse markers (e.g., *"you know", "like"*), editing terms within speech repairs (e.g., *"I mean"*), and filled pauses. A decision tree determines the linguistically motivated sentence boundaries, both within a turn and between two turns for the same speaker. The best decision tree yielded a 3.6% error rate for sentence boundary detection. For disfluency detection, a simple repetition detection script was used to find repeated sequences of up to four words. A shallow chunk parser was also used to support the decision tree's detection of false starts. Zechner evaluated false start detection on 3,000 sentences of the Switchboard corpus using the human transcriptions and obtained an F-score of 0.61 for false start detection and 0.96 for non-false-start detection.

Lendvai et al. [54] used a memory-based learning algorithm to detect disfluencies in Dutch. The definition of disfluencies is fairly broad in this study, including everything that does not fit in the tree structure of a sentence. They used a manually built syntactic tree for each utterance, from which the reference tag for each

[5]Although this work also detects sentence boundaries, we postponed its description until now so that we can discuss sentence boundary and disfluency detection results together.

word (inside or outside a disfluency) is obtained. The features they used include lexical features, word, word context, and overlap within windows (not POS tags). Infrequent or unknown words were processed using an attenuation method. A 97% accuracy (F-measure 80.0%) was obtained on a small corpus of Dutch spontaneous speech using reference transcriptions.

Stolcke and Shriberg [55] and Heeman and Allen [56] both extended the traditional N-gram language model to deal with sentences that include repairs. Stolcke and Shriberg [55] incorporated disfluency resolution into a word-based language model with the assumptions that probability estimates for words after a disfluency are more accurate if conditioned on the intended fluent word sequence and that disfluencies themselves can be modeled as word-like events, each having a probability conditioned on its context. In predicting a word, they summed over the probability distributions for each type of repair (including no repair at all). For hypotheses that include a repair, the prediction of the next word was based on a cleaned-up representation of the context, taking into account whether a single or double word repetition was predicted. They found that on the Switchboard corpus the model reduced word perplexity only in the neighborhood of disfluency events; however, overall differences were small and had no significant impact on recognition accuracy. The goal of such a disfluency LM is to improve speech recognition accuracy by better modeling the occurrence of disfluencies in spontaneous speech.

Heeman and Allen [56] proposed a statistical language model which includes the identification of POS tags, discourse markers, speech repairs, and intonational phrases. An example of such a tightly coupled LM is shown in Equation (2.1). A is the acoustic signal; W, D, R, E, and I represent the word sequence, POS sequence, repair annotation sequence, editing term sequence, and intonational phrase sequence, respectively. The speech recognition problem is redefined so that its goal is to find the sequence of words and the corresponding POS tags, intonation, editing terms, and repair tags that are most probable given the acoustic signal.

$$\begin{aligned}\widehat{WDREI} &= \arg\max_{WDREI} Pr(WDREI/A) \\ &= \arg\max_{WDREI} Pr(A/WDERI)Pr(WDREI) \qquad (2.1)\end{aligned}$$

The second term is the language model probability and can be rewritten as follows.

$$\begin{aligned}\Pr(W_{1,N}D_{1,N}R_{1,N}E_{1,N}I_{1,N}) &= \prod_{i=1}^{N} Pr(W_iD_iR_iE_iI_i/W_{1,i-1}D_{1,i-1}R_{1,i-1}E_{1,i-1}I_{1,i-1}) \\ &= \prod_{i=1}^{N} Pr(I_i/W_{1,i-1}D_{1,i-1}R_{1,i-1}E_{1,i-1}I_{1,i-1}) \\ &\quad Pr(E_i/W_{1,i-1}D_{1,i-1}R_{1,i-1}E_{1,i-1}I_{1,i}) \\ &\quad Pr(R_i/W_{1,i-1}D_{1,i-1}R_{1,i-1}E_{1,i}I_{1,i}) \\ &\quad Pr(D_i/W_{1,i-1}D_{1,i-1}R_{1,i}E_{1,i}I_{1,i}) \\ &\quad Pr(W_i/W_{1,i-1}D_{1,i}R_{1,i}E_{1,i}I_{1,i}) \qquad (2.2)\end{aligned}$$

Equations (2.1) and (2.2) were also extended to include more information, such as the correction of repairs. Experimental results show that when the extended LM is applied to human transcriptions of the Trains corpus, it is able to identify 72% of turn-internal boundaries with a precision of 71%, and 97% of the discourse markers with 96% precision, while detecting and correcting 66% of repairs with 74% precision. They attribute these results to their LM that accounts for the interaction of the tasks of identifying intonational phrases, discourse markers and POS tags, and detecting and correcting speech repairs. Heeman and Allen [56] also pointed out that a prosody model could be integrated with the LM approach as shown in Equation (2.2).

Current speech recognizers rely upon LMs for resolving acoustic ambiguity. Disfluencies are often not handled specially, i.e., they are only captured by the N-gram word sequence. LMs with the ability to model disfluencies [55, 56] seem promising. However, Stolcke and Shriberg's LM actually increases perplexity and word error rate on the Switchboard corpus. They claimed the reason for this is that disfluencies

are inherently a local phenomena that are modeled surprisingly well by standard N-grams, even without context "cleanup". They also attributed their results to their treatment of filled pauses: utterance-medial filled pauses should be cleaned up before predicting the next word; whereas, utterance-initial ones should be left intact. Heeman and Allen's LM allows the speech recognizer to model other aspects of the speaker's utterances, in addition to the words, and thus generates more structural information of the speaker's turn for later processing. The model is able to capture the interactions that exist between word prediction and a variety of other phenomena in dialogue. However, joint modeling of words and tags in the LM could increase the sparse data problem. Also such a LM needs to be trained from annotated training data, whose size is generally much smaller than the text corpus used to train a traditional word-based LM. It is important to note that Heeman and Allen's tightly coupled LM was only evaluated on the Trains corpus, which has a more constrained grammar than general spontaneous conversational speech and thus is relatively easier to model. Additionally, investigation of the performance of such a tightly coupled LM on ASR output remains to be tested.

Combining Textual and Prosodic Information For Disfluency Processing

Text-based approaches have largely left open the question of whether there exist effective acoustic and prosodic cues for repairs. Different studies have been conducted in an attempt to answer this question.

Some studies from linguistics suggest that prosodic emphasis is not utilized in human perception of the disfluencies. Fox Tree [45] found that listeners could not reliably detect where the edits occurred when they were provided with the speech that was edited to remove false starts and repetitions, implying that the prosodic cues alone were not perceptible. However, some prosodic cues may exist by comparing the reparandum and correction regions. Levelt and Cutler [57] found that 55% of repairs involving erroneous words were not prosodically emphasized. Cutler [58] in a

different corpus, found that 62% of speech repairs were not prosodically emphasized. Repairs that are used to restructure the speech but not necessarily make lexical changes are even less likely to be prosodically emphasized, in fact, 81% of them were not prosodically marked. Repairs involving phonetic errors were never prosodically emphasized. Most of these studies from linguists are focused only on the prosodic emphasis in the corrections of disfluencies. Additional acoustic properties still remain to be investigated.

Lickley et al. [59] found that listeners do not detect a disfluency at the point of interruption, but at a point later in the speech stream. In another study, Lickley and Bard [60] used a word gating paradigm[6] to discover how much information is necessary for detecting a disfluency. They found that nearly 80% of the disfluencies in their corpus were detectable at the first word gate in a correction, sometimes even before lexical access of that word had occurred. Their results suggest that acoustic cues at the interruption points or before are insufficient to detect disfluencies and that accessing some information after the interruption point is necessary.

Hindle [61] originally suggested that an "edit signal" serves as a cue that fluent speech has been interrupted. Although no evidence for a single such cue has been found, several corpus studies have found combinations of cues that could be used by algorithms to identify disfluencies with reasonable success. Some potentially useful prosodic cues include: glottalization at the end of a reparandum especially in vowel final fragments [13, 49], silence duration and prepausal lengthening [62], differences in F0 across the interruption point [13, 63], presence of similar F0 contours in the reparandum and the correction [62,64], and the juncture properties between the offset of the reparandum and the onset of the correction that differ from fluent boundaries due to their lack of coarticulation [5].

Nakatani and Hirschberg [13] proposed that speech repairs should be detected in a "speech-first" model using acoustic-prosodic cues, relying less on a word transcription. They developed the "repair interval model" to provide a general model

[6]Stimuli were presented in chunks, which increase by one word each time.

of the temporal intervals that comprise a repair and explored a variety of acoustic and prosodic signals associated with the regions of these intervals. They used hand-transcribed prosodic-acoustic features such as silence duration, energy, and pitch, as well as traditional text-first cues, such as the presence of word fragments, filled pauses, word matches, word replacement, POS tags, and the position of the word in the turn. A decision tree was built to carry out the classification. They obtained a detection recall rate of 86% with a precision of 91% using a portion of the ATIS corpus. Note that their corpus contained many word fragments and their training and testing included only turns with speech repairs; hence, their "findings should be seen more as indicative of the relative importance of various predictors of speech repair location than as a true test of repair site location" [13].

O'Shaughnessy [65] examined the acoustic aspects of false starts[7] and conducted experiments on their automatic identification. He found that most disfluencies occurred in the middle of an utterance and were accompanied by silent pauses of 100-400 ms; whereas, a minority of disfluencies occurred within the first three syllables of the utterance and had a variable amount of pausing. Acoustic analysis showed that when a word was repeated, in most cases it had virtually the same prosodic features in both its instances and that there were a number of times where the repeated word had shorter duration and lower pitch. However, when there is a revision, the second instance had a greater amount of stress. O'Shaughnessy conducted automatic detection experiments on the ATIS corpus. When he used the simple rule of "pause $<$ 400 ms, then disfluency", 70% of disfluencies were correctly identified, with 35% false alarms. When some spectral analysis was exploited for repetition detection, the results were described as being even better, although specific numbers were not reported.

Using an approach similar to the one used for sentence boundary detection [12], Shriberg and Stolcke [18] conducted experiments on disfluency detection using a prosody model, a language model, and their combination on the Switchboard cor-

[7] "False starts" in this work are edit disfluencies.

pus. They found that a prosody model alone (using features such as pause length and F0) performed significantly better than chance performance, and also outperformed a language model on detecting false starts. However, experiments were evaluated only on downsampled data in that research rather than the true test set. In another investigation on non-downsampled Switchboard data [30], Stolcke et al. reported results on the overall accuracy of event detection, including sentence boundaries, fillers, and edit disfluencies. They found that the prosody model was significantly better than chance and that the combination of the prosody model and LM outperforms either model alone. Because the system performs better on sentence boundary detection than on disfluencies and sentence boundaries are more frequent than disfluencies, the overall result does not represent the system's performance on disfluency detection.

Snover et al. [66] used a transformation based learning (TBL) for the detection of disfluencies, under the assumption that the majority of disfluencies can be detected using lexical features, without the use of many prosodic cues. TBL rules were learned from features including lexeme, POS tags, whether the word was followed by a pause, and whether the lexeme was used more often than the average by the speaker. Only very simple prosodic features are used in this algorithm, i.e., pause information after each word. This approach was applied to both the reference transcriptions and the recognition output of the CTS corpus in the 2003 NIST disfluency detection task. The results are worse than the other systems in the evaluation that used both the prosodic and textual knowledge sources. However, since the methods that are used to model textual information are different across systems, it is hard to say whether the poorer performance is due to the limited prosodic features used in this system.

2.2.3 Summary of Past Research on Disfluencies

Studies from linguistics and psychology have shown that disfluencies play an important role in a speaker's utterance planning as well as in discourse and that some disfluencies (e.g., repetitions) are often not detrimental to listeners' comprehension.

Disfluencies are not random phenomena; instead, some measurable patterns exist in disfluencies [41]. Understanding possible disfluency patterns can help us to build a better automatic disfluency detection model.

Prior research has shown some progress on automatic disfluency detection by using prosodic cues, statistical LMs, and syntactic structure information. Some related research on disfluency detection is summarized in Table 2.2. It is worth pointing out that in most of the prior work, sentence boundary information is available for disfluency detection, which makes it a relatively easier task compared to without accessing sentence boundary information. Notice that most research has been conducted using reference transcriptions. The research on disfluency detection is still at an early stage. Clearly investigations using reference transcriptions can provide useful ideas for better modeling spontaneous speech; however, when testing on ASR output, the presence of incorrect words is likely to create serious problems, since all the information extracted from the words or at the syntactic level will be less reliable. This suggests that we need to develop a better understanding of the interaction between disfluency detection and speech recognition.

2.3 Filler Word Processing

Filler words are treated as a category in structural event detection tasks; therefore, we put the prior work related to filler word processing in a separate section, even though they are sometime included in "disfluencies" in some of the prior studies for disfluencies, as have been already discussed in Section 2.2.

2.3.1 Production and Perception of Fillers

Filled pauses (FPs) are defined as vocalized hesitation in the flow of an utterance. A filled pause may occur when a speaker needs to think about what to say next. The speaker actually interrupts his or her speech while continuing his or her articulation. This articulation is not considered to be a word by many people. FPs

Table 2.2
A summary of some important prior studies on disfluency detection. Column two is the taskfor each investigation; column three describes the model or information sources used by each investigation; column four is the corpus on which the experiments were conducted; column five represents whether the experiments were performed on human transcriptions (Ref) or recognition results (ASR). * In Core [52], preliminary repair information is provided, and the parser further corrects them.

Investigation	**Tasks**	**Model**	**Corpus**	**Ref or ASR**
Shriberg, Stolcke et al. [18,30]	IP detection	prosody, word-LM	CTS	Ref, ASR
Bear et al. [49]	repairs	disfluency pattern, parser	ATIS	Ref
Charniak, Johnson [50,51]	repairs	textual information, parser	CTS	Ref
Core [52]	repairs	parser*	Trains	Ref
Nakatani, Hirschberg [13]	repairs	prosody, textual information	ATIS	Ref
Heeman, Allen [56]	repairs and other structural information	LM	Trains	Ref
Snover et al. [66]	repairs and IP	lexical features using TBL	CTS	Ref, ASR

serve an important purpose in helping a speaker hold the conversational floor. When a speaker appears to have finished an idea but wishes to continue speaking, although a subsequent utterance is not yet prepared, an FP may be uttered in order to keep control of the conversational floor. Note that although some word lengthening plays a similar role to a filled pause, this phenomenon is not categorized as a filled pause in this research.

As pointed out previously in Chapter 1, filled pauses and discourse markers can act as an editing term in a disfluency. For example, "*I* ⟨*uh*⟩ *I like it, I* ⟨*you know*⟩ *I like it*". Levelt [67] found that 62% of repairs in a corpus of spontaneous task-oriented utterances included some type of editing expressions, among which "*uh*" was the most common one.

From a different perspective, Fox Tree [45] found suggestive evidence that fillers (meaning FPs in that research) affect comprehension, and that the two fillers *"um"* and *"uh"* affect listeners' comprehension differently. *"Ums"* seem to help comprehension, possibly by providing information about the meta-communicative process, such as directing listeners' attention to the upcoming phrase. *"Uhs"*, in contrast, had no effect on word recognition, perhaps because the effects were masked by pausing effects.

Brennan [44] showed in her experiments that interruptions marked by a filler are better error signals than interruptions without them. This follows Levelt's proposal that an editing expression like *"uh"* may "warn the addressee that the current message is to be replaced" [38]. She also showed that it is not the phonological form of the filler that was driving the fast comprehension, but the extra time that elapsed before the corrections due to the presence of fillers. A longer editing interval gives the listeners more time to process the evidence that there is some trouble, and thus listeners are able to better process the disfluencies.

2.3.2 Past Research on Filler Word Processing

Some HMM-based recognizers (e.g., [68]) regard filled pauses as out-of-vocabulary words and deal with them by using a subword-unit based decoder for processing unknown words. Currently most recognition systems include filled pauses in their vocabulary and build acoustic models for them, thus making filled pause detection in the subsequent structural event detection system essentially a word spotting task.

Goto et al. [69] proposed a method that detects filled pauses and word lengthening on the basis of small fundamental frequency transitions and small spectral envelope deformations under the assumption that speakers do not change articulator parameters during filled pauses. A recall rate of 84.9% and precision of 91.5% were achieved on a Japanese spoken dialogue corpus.

Siu and Ostendorf [70] created a LM to account for three roles a filled pause can take on, namely, when it is utterance initial, part of a repair, or simply a filled pause to hold the floor. By allowing each of these roles to be distinguished, they were able to reduce the perplexity of their LM. They also observed a reduction of LM perplexity for a few discourse markers, such as *"you know"*. Whether the perplexity reduction of their LM can transfer to word error rate reduction in speech recognition is currently untested.

Filled pauses and discourse markers are strongly dependent on word identity. A simple approach for filler word detection is to use lexicon lookup; however, to determine whether the potential words are really fillers, especially in the case of discourse markers, is not a straightforward problem due to their role ambiguity. Sometimes even humans have difficulty judging their role in the face of ambiguity. More knowledge about the dialog is needed to better model discourse markers. Additionally, word errors in ASR output (especially in the filler words themselves) can seriously affect their detection.

2.4 Chapter Summary

We have described some work from psychologists and linguists on the production of disfluencies, the role of fillers, and the effect of these phenomena on listeners' comprehension. We believe that the insights from these efforts will help us to build better automatic detection models. Using an analog from speech recognition, understanding how speech is perceived has proven extremely beneficial to the construction of better automatic speech recognition systems.

Past work has shown that both textual and prosodic cues provide important information for the detection of sentence boundaries and disfluencies. Potential prosodic cues include pauses, word or syllable lengthening, F0 features, interrupted words, glottalization, laryngealization, and increased stress on a correction word versus a reparandum word. Note that we have only focused on prior work related to detecting sentence boundaries and disfluencies and did not introduce all prior work related to prosody. Prosody has been extensively investigated to derive more information about discourse structure. In the speech-to-speech translation system in the Verbmobil project [14], prosody (including F0 and duration) was used to guide the rescoring of an N-best list of word hypotheses produced by a speech recognizer.

Much of the previous work on sentence boundary and disfluency detection has been conducted only on human transcriptions. Although the study of human transcriptions can shed some light on potential features and useful models, our final goal is to enrich speech recognition results, which contain a variety of errors and thus make detection more difficult. Language models or other machine learning techniques that capture textual information are effective, but they will be seriously affected by the word errors in ASR outputs. Without correct word identity, fillers cannot be found, disfluency patterns may not be matched, and sentence-initial word information may be corrupted. It is important to begin looking at the impact of speech recognition on structural information extraction.

Even though previous research is divided into different categories based on the target event type, this does not imply that each problem must be addressed separately. For example, fillers at the start and the middle of a sentence have different impacts on a LM [55, 70], and many prior studies on disfluency detection rely on the availability of sentence boundary information. Future work should take into account the interactions among different structural events.

Much work remains to be done on structural event detection for rich transcription of speech to generate more readable recognition output or help downstream processing modules. The approach that will be employed in this thesis builds upon Shriberg and Stolcke's framework for sentence boundary and disfluency detection [12, 18]. As described in Chapter 1, we will build a more robust prosody model, utilize lexical and syntactic information, and more optimally combine different knowledge sources.

3. DATA RESOURCES AND TASKS

There currently exist two corpora with different speaking styles that were annotated by LDC with structural events using annotation guidelines [71] for the DARPA EARS program [72]. The ready availability of this data makes it possible for others to compare to our systems; hence, these corpora have been chosen for our investigations of structural event detection. Details about the event types investigated, and the tasks and corpora in the EARS program, are described in this chapter. The experiments in Chapters 5 through 8 use data annotated according to the annotation guideline Version 5.0 [71], but those in Chapter 9 use data annotated with guideline Version 6.2 [73].

This chapter is organized as follows. Section 3.1 describes the structural event types in speech that are investigated in this thesis. Section 3.2 describes the structural event detection tasks and the performance measures for these tasks. Section 3.3 describes the two corpora that are used for the experiments.

3.1 Structural Speech Events Types

There are many types of markups that can make a transcription more readable, for example, the addition of speaker turn information. The events this thesis investigates are structure-oriented, including the identification of sentence boundaries, fillers, and edit disfluencies. Each type is described briefly in the next three subsections.

Table 3.1
Structural events annotated by LDC and investigated in this thesis. Note that the subtype of an edit disfluency is not annotated LDC, nor is the correction in an edit disfluency.

Event Type	Subtype	Annotation Mark
SU	Statement	./
	Question	?/
	Backchannel	@/
	Incomplete	.../
Filler	Filled pause	⟨ ⟩
	Discourse marker	
	Explicit editing term	
Edit disfluency	Repeat	[(original utterance) * ⟨editing term⟩ correction]
	Revision	
	Restart	
	Complex	

3.1.1 Sentence-like Units (SUs)

"Sentences" in spontaneous conversational speech can be quite different from those in written text or read speech. Section 4 of the structural event annotation guideline [71] calls these sentence-like units **SUs**. SUs express a speaker's complete thought or idea. Often times this unit corresponds to a sentence, other times to a unit that is semantically complete but smaller than a sentence (e.g., a noun phrase in response to a question). An SU boundary usually coincides with a syntactic clause boundary, but this is sometimes not the case. An SU can be an entire well-formed sentence, phrase, or a single word. Many short SUs are not full sentences or clauses but are nevertheless complete units. Given the differences between sentences and SUs, we use this term from now on to indicate the sentence-like units in conver-

sational speech. There are four types of SUs that are annotated using the LDC annotation guideline [71]:

- Statement: A complete SU that functions as a declarative statement. For example, "*Money seems to be too big of an issue with what's going on today./*" in the example of Chapter 1 on page 3. Short phrases that are not grammatical can be a complete statement SU, for example, as a response to a question.

- Question: A complete SU that functions as an interrogative (including both wh and yes-no questions). Note again that a speaker can use statements with a rising final intonation to make an utterance act as a question, for example, "*And that's a possible next step ?/*" on page 7.

- Backchannel: Sometimes called an acknowledgment or continuer, a backchannel is a word or phrase that encourages the dominant speaker to continue talking by indicating that the non-dominant speaker is still listening to the conversation. Backchannels serve a function similar to gestures like head nodding. Examples of backchannel words include *hm, hmm, right, huh, sure, mm-hm, yeah, oh, yep, okay, yes, really, uh-huh.*

- Incomplete SU: This occurs when a speaker is interrupted and then does not continue with the old utterance or when a speaker trails off. For example, "*and get down to the values of you know i mean .../*" in the example of Chapter 1 on page 3.

3.1.2 Fillers

As indicated in Section 2 of the annotation guideline [71], fillers include filled pauses, discourse markers, and explicit editing terms.

Filled pauses (FPs) are non-lexemes (non-words) that speakers employ to indicate hesitation or to maintain control of a conversation while thinking about what to say

next. FPs can occur anywhere in a speech stream. FPs in the current annotation guideline [71] are limited to the following words: *ah, eh, uh, um.*

A discourse marker (DM) is a word or phrase that functions primarily as a structuring unit of spoken language. It frequently appears at the beginning or the end of an SU. To the listener, it signals the speaker's intention to mark a boundary in discourse, including a change in dominant speaker or the beginning of a new topic. The following words are often used as DMs: *actually, now, anyway, see, basically, so, I mean, well, let's see, you know, like, you see.*

The third filler type is explicit editing term. These are editing terms that are not defined as filled pauses or discourse markers. For example, in *"Today is (Monday), ⟨sorry⟩, Tuesday"*, *"sorry"* is an explicit editing term. Filled pauses and discourse markers can also function as an editing term within an edit disfluency, as in the following example (inside brackets ⟨ ⟩):

(with) ⟨uh⟩ with (many) many people

However, these are not marked as explicit editing terms.

3.1.3 Edit Disfluencies

Edit disfluencies happen when people need to either refocus or revise what they are saying. As indicated in Section 3 of the annotation guideline [71], edit disfluencies follow a basic pattern, each part of which is described below. In the examples shown in this thesis, the extent of an edit disfluency is marked with square brackets [ ]. Note that in the annotated data in the EARS program, the correction part is not annotated, nor are the subtypes of edit disfluencies; however, this information is provided in examples in order to better illustrate the structure of an edit disfluency.

Edit disfluency template (appears embedded in a sentence):

[(original utterance) * ⟨editing term⟩ correction]

- Original utterance: Also called a "reparandum," this is the portion of the utterance that is corrected or abandoned entirely (in the case of restarts). This portion would be discarded when removing disfluencies for a cleaned-up transcription. As shown in the template, the original utterance is indicated by parentheses. This convention was used in the example in Chapter 1 on page 3 and also in the examples provided in this section.

- Interruption point (∗): This is the point at which the speaker breaks off the original utterance, and then repeats, revises, or restarts his or her utterance. Interruption points are marked with '∗' in our examples.

- Editing term: Some edit disfluencies include an overt statement from the speaker marking their existence. The term can consist of a filled pause, a discourse marker, or an **explicit** editing term (such as *"sorry", "excuse me"*). An editing term is optional in an edit disfluency.

- Correction: This consists of the portion of the utterance that corrects the original utterance. It is the part that would remain after the cleanup of the transcription.

Based on their internal structure, edit disfluencies can be divided into the following four subtypes:

- Repetitions: The speakers repeat some part of the utterance. For example, [**(we may not) * we may not**] *have as high a standard of living.*

- Revisions (content replacements): The speaker modifies the original utterance using a similar syntactic structure. For example, *Show me flights* [**(from Boston on) * ⟨uh⟩ from Denver on**] *Monday.*

- Restarts (false starts): A speaker abandons an utterance or a constituent and then starts over entirely. For restarts, the correction region in the disfluency template above is typically marked as empty. For example, [**(It's also) * **] *I used to live in Georgia.*

- Complex disfluencies: A speaker produces a series of disfluencies in succession or in nested structure. For example, **[(I think * I * I) * ⟨now⟩ I think]** *people generally volunteer*. The internal structure (nested or sequential structure) inside the complex disfluency is not annotated based on the guideline [71]; however, the internal interruption points are marked. This annotation is different from the disfluency annotation in the Switchboard Penn Treebank data [19], which indicates the internal structural information of disfluencies, as shown below:

  [[[I think * I] * I] * ⟨now⟩ I think] people generally volunteer

  where each square bracket represents a disfluency with the reparandum and correction regions split by the IPs (*). Nested structure can be represented using this annotation scheme.

3.2 Structural Event Detection Task Description

Our focus is on the official Rich Transcription structural metadata extraction (MDE) tasks defined in the DARPA EARS program due to the availability of the annotated data that can be used for system training and testing, as well as the availability of scoring tools. We believe the methodology developed for these tasks can generalize to other structural event detection tasks or other similar speech and language processing tasks.

3.2.1 Task Description

The 2003 Rich Transcription (RT-03) structural MDE task includes four subtasks, described below.

- SU boundary detection: The goal is to find the end point of an SU. Note that SUs may correspond to either complete or incomplete utterances.

- Filler word detection: The goal is to identify words used as filled pauses (FPs), discourse markers (DMs), or explicit editing terms (EETs) in edit disfluencies.
- Edit word detection: The goal is to find all the words within the reparandum region of an edit disfluency. This is essentially the portion of an utterance that if deleted results in a more "fluent" version of the utterance.
- Interruption point (IP) detection: The goal is to find the interword location at which point fluent speech becomes disfluent. In addition to the IPs within the edit disfluencies, IPs defined in RT-03 include the boundary before filler words.

Note that it is convenient to divide tasks into separate subtasks, which for example makes scoring easier. However, these subtasks are not independent. In fact edit word detection and IP detection are clearly interdependent. Additionally there are some ambiguities across different event types (e.g., an incomplete SU versus restart edit disfluency), which will affect multiple subtasks.

Among the tasks above, there are two types: boundary detection and extent detection tasks. SU boundary detection and IP detection belong to the boundary detection category;[1] whereas, filler word and edit word detection tasks involve extent detection. The boundary detection task is equivalent to a classification task, i.e., for each interword boundary, a decision is made about whether there is a structural event at that position. The extent detection task, on the other hand, needs to determine a portion of the utterance that is a filler word phrase or the reparandum of an edit disfluency.

Structural event detection is evaluated on two different corpora: conversational telephone speech (CTS) and broadcast news (BN) speech. Details about these two corpora are provided in Section 3.3 of this chapter.

For EARS MDE evaluation, two different types of transcriptions are used: human-generated transcription (REF) and speech recognition output (STT). Using the ref-

[1]The SU detection task is really an extent detection task; however, under the assumption that the end of a previous SU indicates the beginning of the following SU (ignoring possible pauses), for simplicity, SU detection task is treated as a boundary detection task.

erence transcriptions provides the best-case scenario for the evaluation of a structural event detection algorithm. Evaluation across transcription types allows us to study the structural event detection tasks both with and without the confounding effect of speech recognition errors.

3.2.2 Performance Measures

Each of these tasks is evaluated separately. There are several performance measures used for evaluating system performance: the NIST official scoring metric and some additional metrics that convey various types of useful information about system performance. We describe all of the metrics here that are used in this thesis, and then describe which metrics are used and why in various experiments. Generally we use the NIST scoring metric for system performance in order to compare with other systems' performance, but choose other appropriate metrics when focusing on a specific aspect of the problem.

- NIST scoring metric. The NIST scoring tools first align the reference and hypothesis words. This is straightforward when evaluating on the human transcriptions since they match exactly. When recognition output words are used, they usually do not align perfectly with those in the reference transcriptions. In this case, an alignment that minimizes the word error rate is used. After word alignment, the hypothesized structural events are mapped to the reference events using the word alignment information, and then unmatched structural events are counted. For edit and filler word detection, the error rate is the average number of misclassified reference tokens per reference edit or filler word token. For SU and IP detection, the error rate is the number of misclassified boundaries per reference SU or IP. For example, the following equations show the NIST error rate for SU detection and edit word detection:

$$\text{SU error rate} = \frac{\text{number of incorrect boundaries}}{\text{total number of SU boundaries}} \tag{3.1}$$

$$\text{Edit word error rate} = \frac{\text{number of misclassified words}}{\text{total number of edit words}} \tag{3.2}$$

The error rate in the NIST metric can be greater than 100%. The following example shows a system SU hypothesis aligned with the reference SUs:

```
Reference:  w1      w2      w3  /   w4
System:     w1  /   w2      w3      w4
                ins             del
```

where w_i is a word and '/' indicates an SU boundary. There are two misclassified boundaries: one insertion error and one deletion error (indicated by 'ins' and 'del') in the example above. Since there is only one reference SU boundary, the NIST SU error rate for this system output is 200%. A detailed description of the scoring tool is provided in http://www.nist.gov/speech/tests/rt/rt2003/fall/. If a system hypothesizes a non-event boundary at each interword boundary, then the NIST error rate will be 100% for the boundary detection tasks, all due to deletion errors, without any insertion errors. This is used as a baseline performance.

- Classification error rate (CER). The structural event detection problem can be treated as a classification problem for which performance can be easily measured using CER. CER is defined as the number of incorrectly classified samples divided by the total number of samples (not just the positive samples). For this measurement, the samples are all the interword boundaries for the boundary detection tasks or the total number of words for the extent detection tasks. In the example shown above, there are four word boundaries, among which two are misclassified, therefore the CER is 2/4=50%. The baseline performance (also called chance performance) using CER is equal to the event prior.

The NIST error rate and CER tend to be highly correlated. When a reference transcription is used, the errors in NIST's metric correspond almost directly to classification errors. The major difference lies in the denominator: the number of reference events is used in the NIST scoring metric; whereas, the total number of word or word boundaries is used in the CER measure. When using recognition output, CER is not well defined due to the presence of insertion and deletion errors in the recognized word stream. However, given the correspondence between NIST error rate and CER for the reference condition, the NIST error rate can be converted proportionately to CER for the STT condition as follows:

$$\text{CER} = \text{NIST error rate} \times \text{Event prior} \tag{3.3}$$

- F-measure. In a classification or detection task, the F-measure is defined as follows:

$$F\text{-}measure = \frac{(1+\beta^2) \times recall \times precision}{\beta^2 \times recall + precision} \tag{3.4}$$

where $precision = \frac{TP}{TP+FP}$, $recall = \frac{TP}{TP+FN}$, and TP and FP denote the number of true positives and false positives, respectively. FN represents the number of false negatives, and β corresponds to the relative importance of *precision* versus *recall*. β is set to 1 if false alarms and misses are considered equally costly. For this measure, the minority class is the positive class, i.e., the SU or IP boundaries, or the filler or edit words.

- The Receiver Operating Characteristics (ROC) and the Area Under the Curve (AUC). ROC curves [74, 75] can be used to enable visual judgments of the trade-off between true positives and false positives for a classification or detection task. Depending on the application, an appropriate operating point from the ROC curve can be selected [76]. For structural event detection tasks, a threshold needs to be selected to minimize the overall classification error rate or NIST error rate. The AUC can tell us whether a randomly chosen majority

class example has a higher majority class membership than a randomly chosen minority class example; thus, it can provide insight on the ranking of the positive class examples. F-measure, ROC, and AUC measures are only used for the reference condition, due to the imperfect alignment problem when using recognition output.

Currently there exist no standard tests for significance test using the NIST scoring method. The problem is that the metric is not based on a consistent segment [77]. For the CER metric, the sign test can be utilized to test significance at the word boundary level. We believe the findings based on the sign test are likely to transfer to other methods for significance test.

3.3 Corpora

Conversational telephone speech (CTS) and broadcast news (BN) are used in the structural event detection tasks in EARS. We believe investigations using these two corpora can enhance our understanding of how structural information is represented in human languages. In CTS, participants are paired by a computer-driven "robot operator" system that sets up the phone call, selects a topic for discussion from a predefined set of topics, and records the speech into separate channels until conversation is complete. Each conversation is about 6 minutes on average. BN contains news broadcasts from ABC, CNN and CSPAN television networks, and NPR and PRI radio networks. Figure 3.1 shows examples of human-generated transcriptions for CTS and BN, respectively. The CTS transcription is diarized because it is a two person dialog.

CTS and BN are very different genres. They differ in both average sentence length and frequency of disfluencies. Speech in BN has fewer disfluencies, sentences tend to be longer and more grammatical, and the speakers are mostly professionals reading teleprompted text. Speech in CTS is more casual and conversational, containing many backchannels, filler words, and edit disfluencies.

CTS:

speaker A: *hi um yeah i'd like to talk about how you dress for work and and um what do you normally what type of outfit do you normally have to wear*

speaker B: *well i work in uh corporate control so we have to dress kind of nice so i usually wear skirts and sweaters in the winter time slacks i guess [noise] and in the summer just dresses*

speaker A: *uhhuh*

speaker B: *we can't even well we're not even really supposed to wear jeans very often*

speaker A: *and is*

speaker B: *so it really doesn't vary that much from season to season since the office is kind of you know always the same temperature*

BN:

the top selling car of nineteen ninety-seven was announced today and the winner is toyota camry toyota out sold both the honda accord and the ford taurus which has been number one for the past five years on wall street today the dow jones industrials lost just under four points to close at seventy-nine oh two on the nasdaq market stocks lost almost eighteen and a half points just ahead is there the possibility of peace between america and iran

Fig. 3.1. Examples of transcriptions for CTS and BN, respectively. SU boundaries are not shown in the examples.

The data used for training and evaluating our structural event detection models is taken from the official NIST RT-03 data. Training and test data were annotated with structural events by LDC using guidelines detailed in [71]. The CTS data set contains roughly 40 hours of speech (377 conversations) for training and 6 hours (72 conversations) for testing. The BN data contains about 20 hours of speech for training and 3 hours (6 shows) for testing. There are about 90 shows in the BN training data; however, for most shows, only some portion is annotated with structural events.

Table 3.2 shows the class distribution of different structural event types in the two corpora, along with the data size, and the WER of the speech recognition output on the test set. WER is determined using the recognition output from SRI's recognizer used in the 2003 NIST evaluation [78].

Table 3.2
Information on the CTS and BN corpora, including the data set sizes, the percentage of the different types of structural events in the training set, and the word error rate (WER) of the speech recognizer on the test set.

	CTS	BN
Training size (number of words)	480K	178K
Test size (number of words)	71K	24K
WER (%)	22.9	12.1
SU percentage	13.6	8.1
Edit word percentage	7.4	1.8
Edit IP percentage	4.5	1.1
Filler word percentage	6.8	1.8
Filled pause percentage	2.9	1.2
Discourse marker percentage	2.5	0.4

Note that the corpora described here are composed of speech and annotated transcriptions (both CTS and BN) that are used for training and testing the structural event detection models. There is much more additional speech data with corresponding transcriptions that is used for training the speech recognition models. Because annotations require much more effort than transcribing speech, the annotated data size is generally much smaller than the size of the data used for training the acoustic and language models for speech recognition. Some additional text corpora will be utilized for language model training in the structural event detection tasks, which will be described in Chapter 6.

4. THE HMM APPROACH TO STRUCTURAL EVENT DETECTION

In this chapter, the HMM is introduced that is used as a baseline method for the boundary detection tasks. This approach builds upon the prior work of Shriberg and Stolcke [12, 31]. This chapter presents the general approach; whereas, the specific models for different event types are discussed in Chapter 5.

This chapter is organized as follows. Section 4.1 provides an overview of the three components of the HMM. Section 4.2 describes the textual and prosodic features used in this system. Models constructed for each knowledge source are introduced in Section 4.3 and model combination is described in Section 4.4. A summary for this chapter appears in Section 4.5.

4.1 Overview

The structural event boundary detection task is represented as a classification task, that is, for an interword boundary, a decision is made about whether or not there exists a structural event at that position. For extent detection, the boundary detection approach is combined with additional knowledge processing.

There are three components in the statistical boundary detection algorithm. Each is described in detail in the indicated sections.

- Feature Processing (Section 4.2): develop an inventory of input features for the statistical classifiers, including prosodic features (e.g., temporal, intonational, and energy features) and lexical features (e.g., word co-occurrence, part-of-speech, or keywords).

- Model Construction (Section 4.3): evaluate the use of a variety of model types that capture information from the various knowledge sources, including the prosody and language models. Each component model can be finely tailored to the data and task.

- Model Combination (Section 4.4): integrate selected model types and knowledge sources. In addition to the HMM-based integration approach, we briefly describe other integration approaches such as a simple interpolation of classifier scores.

4.2 Feature Types

4.2.1 Prosodic Features

Prosodic features reflect information about temporal, intonational, and energy contours. Figure 4.1 shows an example of a waveform, with the corresponding pitch and energy contour, the word alignment, and SU boundary information for the utterance *"um no I hadn't heard of that."* Although only word alignment information is shown in the figure, phone-level alignment is also used for prosodic feature computation.

Duration, pitch, and energy features are included in our prosodic feature set. All of these features are associated with each interword boundary and they can be automatically extracted from the word and phonetic alignments of the word sequence. Below is a description of the prosodic features that are investigated and how they are computed. A comprehensive listing of the features in the prosodic feature set, 101 in total, can be found in [79].

- **Duration Features**

  Pause duration after each word boundary is extracted based on the alignment of human transcriptions or recognition output. We also included the duration of the pause preceding the word before the boundary to reflect whether speech

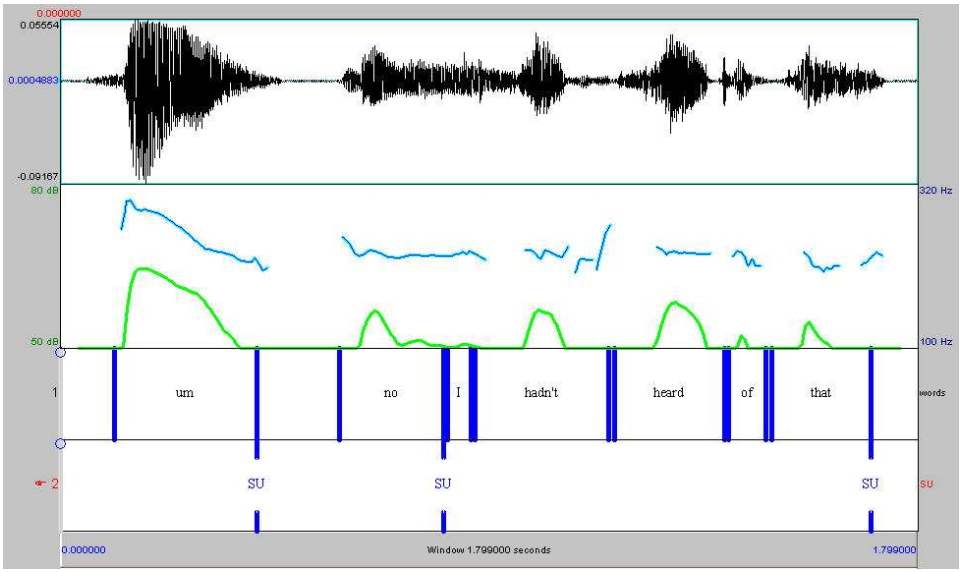

Fig. 4.1. The waveform, pitch and energy contours, word alignment, and SU boundaries for the utterance *"um no I hadn't heard of that"*.

right before the boundary is just starting up or is a continuation of previous speech. Phone durations are also computed. To capture preboundary lengthening, which typically affects the nucleus and coda of syllables, we measure vowel and rhyme duration. For example, the normalized vowel duration is calculated as follows:

$$Norm\_dur_i = \frac{(dur_i - \mu_i)}{\sigma_i} \tag{4.1}$$

where dur_i is the duration for a vowel, and μ_i and σ_i are the mean and the standard deviation of this vowel over all the training data. Index i is used here to represent a vowel. We also extract features such as the duration of the last vowel or the stressed vowel in a multisyllabic word, as well as their normalization. The duration for the word preceding a boundary and its normalization are also included as duration features.

- **F0 Features**

An autocorrelation-based pitch tracker (get_f0 function in the ESPS package) is used to calculate frame-level F0 estimates. These raw F0 values are then post-processed to account for tracking errors, to use speaker dependent parameters, and to simplify the F0 features. For each speaker, the F0 distribution is fitted to a lognormal tied mixture model (LTM) [80], whose mixture weights are found using an expectation maximization (EM) algorithm. The model returns a pitch baseline value for a speaker, which represents the lowest non-halved pitch value that will be used later for pitch normalization. A median filter is also applied to smooth voicing onsets for which the pitch tracker is unreliable. The frame level F0 values are then stylized to simplify tonal contours, shapes, and slopes. A piecewise linear fit (PWL) algorithm [80] is used to create line estimates for the median-filtered F0 values. On a particular voiced region, the PWL algorithm attempts to fit lines by minimizing the mean squared error between the linearized pitch estimates and the raw F0 values using a greedy algorithm. After picking the best fit nodes, the pitch contour is represented as the summation across all of these nodes for the voiced region:

$$F_0 = \sum_{k=0}^{K} (a_k F_0 + b_k) I_{[x_{k-1} < F_0 < x_k]} \tag{4.2}$$

where a_k and b_k are the best parameters chosen by the PWL algorithm for a node indexed by k. K is the total number of nodes in the voiced region. Figure 4.2 shows an example of the raw and the stylized F0 contour for the utterance *"um no I hadn't heard of that"*. As can be seen from the figure, the stylized F0 contour captures the overall pitch contour, at the same time eliminating the effect of imperfect estimation of the raw F0.

Using the stylized pitch contour, several different types of F0 features are computed.

Range features: These features reflect the pitch range of a single word or window relative to the speaker-specific baseline F0 value computed in the LTM model. Examples of such range features are the minimum, maximum, mean,

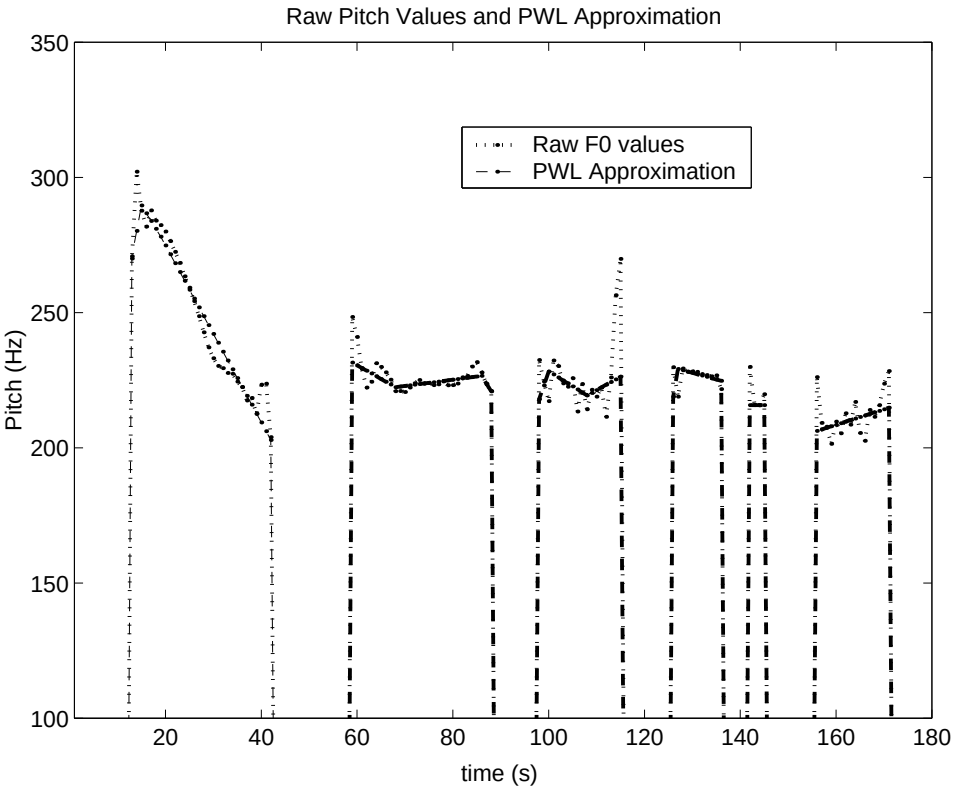

Fig. 4.2. The raw and stylized F0 contours for the utterance *"um no I hadn't heard of that"*.

and last F0 values for each word boundary, excluding values which are unvoiced, halved, or doubled. These features are normalized by the baseline F0 values using a linear difference, log difference, and log ratio. We expect speakers to be more likely to fall closer to the bottom of their pitch range at a phrase, sentence, or topic boundary.

Movement: We take measurements from the stylized F0 contours for the voiced regions of the word preceding and the word following a boundary. The minimum, maximum, and mean F0 values, and the starting or ending stylized F0 values are computed and compared to that of the following word. Log difference and log ratio normalization values are also calculated.

Slope features: The stylized pitch values generate pitch slope within a word or a predefined length of window. The slope across a boundary is compared to capture local pitch variation. A continuous trajectory is more likely to correlate with non-boundaries; whereas, a broken trajectory tends to indicate a boundary of some type.

- **Energy Features**

Speakers tend to start an utterance loudly and taper off over time. Thus we first generate the frame level root mean square (RMS) energy values (obtained from the get_f0 function of the ESPS package), and then compute the minimum, maximum, and the mean RMS values over the word and over the voiced frames. Similar to stylized F0 processing, the raw energy values are fit to a linear model to capture the slope change of energy, and the difference of energy values across a word boundary is also computed.

- **Additional Features**

Some additional automatically extracted features are also included, such as turn-related features and gender features. Like all the prosodic features described above, these features can be automatically extracted from the speech data, using gender detection or automatic speaker segmentation and clustering techniques. These features may interact with the aforementioned prosodic features (e.g., F0 features), so these features are put in the prosodic feature category to model possible interactions. Turn-related features include whether or not there is a speaker change at a boundary, the time elapsed from the start of a turn, and the turn count within the current conversation. Note that gender detection and speaker segmentation algorithms are not accurate, and thus these additional features are imperfect.

4.2.2 Textual Features

Textual information can be represented by lexical features for a word, the co-occurrence of two or more words, a word's part-of-speech tag, or its semantic class. As discussed in Chapter 3, some words are highly correlated with backchannels, filled pauses, and discourse markers. Some cue words and their associated event types are listed in Table 4.1. Although these provide important lexical cues for structural event detection, many are ambiguous (e.g., '*right*' may be used in contexts other than backchannels, '*like*' can be a verb rather than a discourse marker).

Our baseline system largely uses word co-occurrence information, in particular, features such as which words tend to precede or follow a structural event type. Investigations incorporating other types of lexical features and syntactic structure information are described in Chapters 6 and 8.

Table 4.1
Examples of cue words that are highly representative of some structural event types.

Event Type	**Example Words**
Backchannel	yeah, okay, right, uhhuh
Filled Pause	uh, um, mm
Discourse Marker	I mean, you know, like, so

4.3 The Models

4.3.1 The Prosody Model

The goal of the prosody model in the structural event detection task is to determine the class membership for each word boundary using the prosodic features. For the baseline experiments, a decision tree classifier [81] serves as the prosody

model for estimating the posterior probability of an event type at a given interword boundary. A decision tree classifier is used because it offers the distinct advantage of interpretability. This is crucial for a baseline system, which is helpful for us to obtain a better understanding of how prosodic features are used to signal various event types, and select or design other useful features. Second, our preliminary studies have shown that a decision tree performs as well as other classifiers, such as neural networks, Bayes classifiers, or mixture models. Third, the decision tree classifier can handle missing feature values, as well as both continuous and categorical features. Fourth, the decision tree can produce posterior probability estimates that can be easily combined with a language model.

During training, the decision tree learning algorithm selects a single feature that has the highest predictive value, i.e., reduces entropy the most, for the classification task in question. The leaves of the tree store probabilities about the class distribution of all the samples falling into the corresponding region of the feature space, which then serve as predictors for unseen test samples. Various smoothing and pruning techniques are commonly employed to avoid overfitting the decision tree model to the training data. We use the CART algorithm for learning decision trees and the cost-complexity pruning approach, both of which are implemented in the IND package [82]. The software offers options for handling missing feature values and is capable of processing large amounts of training data. On the test set, the decision tree can generate posterior probabilities for each sample representing the likelihood of each class given the prosodic features.

An example of a decision tree is shown in Figure 4.3. This is the decision tree created for the SU detection task on the BN corpus. The features used in this tree are described in Table 4.2. The prosodic features used by the decision tree are associated with a word, and the boundary for which the system needs to make an event-type decision is at the end point of that word.

```
PAU_DUR < 4.5:  0.8651 0.1349 0
|  F0K_WRD_DIFF_LOLO_N < -0.035966:  0.7872 0.2128 0
|  |  F0K_LR_LAST_KBASELN < -0.070707:  0.604 0.396 0
|  |  |  PATTERN_BOUNDARY in rr,ff,fr,rf,rX,fX :  0.5585 0.4415 0
|  |  |  |  LAST_RHYME_DUR_PH_ND_bin < -0.5:  0.7114 0.2886 0
|  |  |  |  LAST_RHYME_DUR_PH_ND_bin < -0.5:  0.7114 0.2886 0
|  |  |  |  LAST_RHYME_DUR_PH_ND_bin >= -0.5:  0.409 0.591 S
|  |  |  |  |  PREV_PAU_DUR < 12.5:  0.3735 0.6265 S
|  |  |  |  |  PREV_PAU_DUR >= 12.5:  0.9269 0.07306 0
|  |  |  PATTERN_BOUNDARY in Xf,Xr :  0.9363 0.06371 0
|  |  F0K_LR_LAST_KBASELN >= -0.070707:  0.8164 0.1836 0
|  F0K_WRD_DIFF_LOLO_N >= -0.035966:  0.9105 0.08952 0
PAU_DUR >= 4.5:  0.1169 0.8831 S
|  PAU_DUR < 16.5:  0.3454 0.6546 S
|  |  TURN_TIME_N < 0.9962:  0.4755 0.5245 S
|  |  |  F0K_DIFF_LAST_KBASELN < 2.3293:  0.2695 0.7305 S
|  |  |  |  PREV_PAU_DUR < 12.5:  0.2439 0.7561 S
|  |  |  |  PREV_PAU_DUR >= 12.5:  0.7222 0.2778 0
|  |  |  F0K_DIFF_LAST_KBASELN >= 2.3293:  0.5828 0.4172 0
|  |  |  |  PREV_PAU_DUR < 7.5:  0.548 0.452 0
|  |  |  |  |  LAST_RHYME_NORM_DUR_PH_ND_bin < -0.25:  0.6467 0.3533 0
|  |  |  |  |  LAST_RHYME_NORM_DUR_PH_ND_bin >= -0.25: 0.4919 0.508 S
|  |  |  |  |  |  F0K_WRD_DIFF_ENDBEG < 0.1034:  0.5456 0.4544 0
|  |  |  |  |  |  |  F0K_WRD_DIFF_LOLO_N < -0.071836:  0.4097 0.5903 S
|  |  |  |  |  |  |  F0K_WRD_DIFF_LOLO_N >= -0.071836:  0.6729 0.3271 0
|  |  |  |  |  |  F0K_WRD_DIFF_ENDBEG >= 0.1034:  0.356 0.644 S
|  |  |  |  PREV_PAU_DUR >= 7.5:  0.8707 0.1293 0
|  |  TURN_TIME_N >= 0.9962:  0.01177 0.9882 S
|  PAU_DUR >= 16.5:  0.07639 0.9236 S
```

Fig. 4.3. An example of a decision tree for SU detection. Each line represents a node in the tree, with the associated question regarding one particular prosodic feature, the class distribution, and the most likely class among the examples going through this node (S stands for SU boundary, and 0 for non-SU boundary). The indentation represents the level of the decision tree. Some of the features used in this tree are described in Table 4.2.

4.3.2 The Language Model (LM)

The role of the LM in speech recognition is to predict the next word given the previous word history; whereas, for structural event detection, the goal of the language model is to capture the structural information (e.g., SUs, disfluencies, and

PAU_DUR	pause duration after a word's end point
LAST_VOW_DUR_Z_bin	binned normalized duration of the last vowel in the word
WORD_DUR	word duration
PREV_PAU_DUR	pause duration before the word
STR_RHYME_DUR_PH_bin	binned normalized duration of the stressed rhyme in the word
TURN_F	whether there is a turn change after the word
F0K_INWRD_DIFF	the log ratio of the first and the last stylized F0 value for the word

Table 4.2
Examples of the prosodic features used for the SU detection problem that appear in the decision tree shown in Figure 4.3.

fillers) contained in the word sequence. For such a goal, a hidden event LM [83] is used to model the joint distribution of boundary types and words in an HMM with the hidden variable in this case being the boundary type. Let W represent the string of spoken words, W_1, W_2, $\cdots$, and E represent the sequence of interword events, E_1, E_2, $\cdots$. The hidden event language model describes the joint distribution of words and events, $P(W, E) = P(w_1, e_1, w_2, e_2, ...w_n, e_n)$. Note that for such a word level LM, no explicit features are used for each word other than word co-occurrence that is directly incorporated in the model.

For training a hidden event LM, hand-labeled data is used such that each event is represented by an additional non-word token that is explicitly included in the N-gram LM. For example, *"I ⟨IP⟩ I like it"*, event ⟨IP⟩ is an additional token in the dictionary. The bigram parameter $P(\langle \text{IP} \rangle | \text{I})$ gives the probability of an IP following the word *"I"*. Note that we do not represent the fluent, intra-sentence boundary events explicitly, considering that they are implied by the absence of other events.

We believe this choice better captures the flow of word strings than including the 'non-event' explicitly in the N-gram LM and avoids fragmenting the training data.

The hidden event LM can be utilized to label a word sequence with the most likely events in an HMM. In this model, the word-event pairs correspond to states and the words to observations, with the transition probabilities given by the hidden event N-gram model. Given a word sequence W, a forward-backward dynamic programming algorithm [84] is used to compute the posterior probability $P(E_i|W)$ of an event E_i at position i. For a boundary detection task, an event $\hat{E}_i$ is found such that it maximizes the posterior probability $P(E_i|W)$ at each individual boundary. This approach also minimizes the expected per-boundary classification error rate.

In addition to a statistical hidden event LM approach, a keyword based language model can be used for detecting fillers and backchannels (see Table 4.1 for examples of such keywords). However, these lexical cues are well captured by a hidden event LM; for example, the bigram probability $P(Type = SU_{backchannel} \mid uhhuh)$ is very high in a hidden event LM, so we choose not to use keyword-based models to locate such structural event types.

4.4 Model Combination

Because prosodic and lexical cues provide complementary information with different levels of granularity, we expect the combination of these knowledge sources will give superior performance over each model alone. Two approaches are described below for model integration:

- Posterior Probability Interpolation

  Since both the prosody model (the decision tree classifier, denote 'DT') and the language model yield posterior probabilities for an event type E_i at each interword boundary, a better estimation of the posterior probability of an event

occurring at that boundary given both the knowledge sources can be obtained by linearly interpolating the posterior probabilities from these two models:

$$P(E_i|W,F) \approx \lambda P_{LM}(E_i|W) + (1-\lambda)P_{DT}(E_i|F), \tag{4.3}$$

where λ can be optimized based on held-out data. $P_{LM}(E_i|W)$ and $P_{DT}(E_i|F)$ are the posterior probabilities generated by the hidden event LM and the prosody model respectively, where W is the word sequence and F represents prosodic features. In addition to combining the prosody model and a word-based LM, this posterior probability interpolation method can be applied to other models, as will be described in Chapter 6.

- An Integrated HMM Approach

  An integrated HMM models the joint distribution $P(W,F,E)$ of word sequence W, prosodic features F, and the hidden event types E in a Markov model. The goal of this approach is to find the event sequence $\hat{E}$ that maximizes the posterior probability $P(E|W,F)$:

$$\hat{E} = \arg\max_E P(E|W,F) = \arg\max_E P(W,F,E) \tag{4.4}$$

  At each position i, the associated prosodic features F_i are modeled as emissions from the hidden states E_i with likelihood $P(F_i|E_i,W)$. Under the assumption that prosodic observations are conditionally independent of each other given the event type E_i and the word sequence W, $P(W,E,F)$ can be rewritten as follows:

$$P(W,E,F) = P(W,E)\prod_i P(F_i|E_i,W) \tag{4.5}$$

  Additionally prosodic observations depend only on the phonetic alignment W_t, ignoring word identity W. This may also make prosodic features more robust

to recognition errors. Therefore, Equation (4.5) can be rewritten using only the phonetic alignment information W_t for the second term:[1]

$$P(W, E, F) = P(W, E) \prod_i P(F_i|E_i, W_t) \tag{4.6}$$

An estimation of $P(F_i|E_i, W_t)$ can be obtained from the decision tree class posterior probabilities $P_{DT}(E_i|F_i, W_t)$ as follows:

$$P(F_i|E_i, W_t) = \frac{P(F_i|W_t)P_{DT}(E_i|F_i, W_t)}{P(E_i|W_t)} \approx \frac{P(F_i|W_t)P_{DT}(E_i|F_i, W_t)}{P(E_i)} \tag{4.7}$$

$P(E_i|W_t)$ is approximated as $P(E_i)$ above, assuming that a structural event is dependent on word identity but independent of word alignment information. Substituting Equation (4.7) and Equation (4.5) into Equation (4.4), we obtain the following expression for the most likely event sequence, using the hidden event LM $P(W, E)$, the decision tree estimation $P_{DT}(E_i|F_i, W_t)$, and the prior probabilities of the events $P(E_i)$:

$$\hat{E} = \arg\max_E P(E|W, F) = \arg\max_E P(W, E) \prod_i \frac{P_{DT}(E_i|F_i, W_t)}{P(E_i)} \tag{4.8}$$

The first term $P(F_i|W_t)$ in the numerator of Equation (4.7) is independent of E, therefore it can be ignored in the *argmax* formula and does not appear in Equation (4.8).

What remains is to explain how $P(E_i|F_i, W_t)$ is calculated during testing. As described earlier, the decision tree prosody model can generate the posterior probability for a test sample. However, if we downsample the majority class in training but apply the trees to non-downsampled test data, there would be a mismatch between the class distribution in the training and test set, and thus the posterior probabilities would need to be adjusted accordingly [85]. For a classification problem, the posterior probability of the class membership for a sample x can be expressed using Bayes theorem:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{4.9}$$

[1] W_t is used here rather than W_{t_i} since F_i is computed using contextual information beyond W_i.

where C_k is the class membership for the sample x. If training and testing sets differ significantly in class distribution, then it is appropriate to use Bayes theorem to make necessary corrections in the posterior probabilities for the test set. This can be done by dividing the output posterior probabilities from the classifier by the prior probabilities corresponding to the training set, multiplying them by the new prior probabilities for the test set,[2] and then normalizing the results.

Notice that the formula above is derived to obtain the most likely event sequence $\hat{E}$. In our system, we use a forward-backward algorithm to find the most likely event for each interword location, rather than using the Viterbi algorithm to determine the most likely event *sequence*. This minimizes the per-boundary classification error rate.

Although there are other alternatives for model combination, for instance, the scores from a LM could be included directly as a feature in the decision tree model, as in [86], results in [12] showed that this approach performs worse than the HMM described in this section. Hence, this method is not considered in this thesis.

4.5 Chapter Summary

In this chapter, we have described the baseline HMM approach for the structural event boundary detection task. At each word boundary, a set of prosodic features is extracted, which reflect the duration, pitch, and energy information. A decision tree is used to implement the prosody model that estimates the event class membership at each word boundary given the prosodic features. A hidden event LM is used to model the joint word and event sequence. These two knowledge sources are integrated in an HMM system. Since effective model combination is very important, alternative methods will be investigated in Chapter 8.

[2]Although the distribution for a test set is usually unknown, it can be estimated from the original non-downsampled training set.

5. HMM BASELINE PERFORMANCE

In this chapter we describe our baseline system's performance for the detection of structural events. Research enhancements will be described in Chapters 6 through 9. For each interword boundary, various knowledge sources are used to determine whether it is a boundary for the structural events of interest, namely SU, disfluency interruption points, or filler word boundaries using the HMM described in Chapter 4. In addition, rule-based knowledge is used for event extent detection (i.e., filler word and edit disfluency detection).

This chapter is organized as follows. Section 5.1 describes our choices of the classes used in the classifiers for the boundary detection tasks, as well as the training and testing procedures. Section 5.2 provides the HMM baseline results for the structural event detection tasks. Section 5.3 summarizes our findings.

5.1 System Description

5.1.1 Choice of Classes

For the boundary detection tasks, a general HMM has been described in Chapter 4. A remaining problem is the choice of target classes used for the system. Recall that we are investigating several structural event detection tasks, SU detection, filler word detection, and edit word and IP detection. For SU detection, since the end of an SU often implies the beginning of a following SU, the SU boundary between words is used to represent the end of an SU and the start of the next one. Because an IP always corresponds to an interword boundary, the selection of a boundary representation for the IP task is obvious.

For filler word detection, the end of a filler word string is used as the class in the boundary detection framework. Since filler words are limited to a word list, knowing the end point of the string allows us to go backward to determine the onset of the filler word sequence. One reason that the end of the filler word string is chosen, rather than its beginning, is that prosodic information should be more helpful for locating the final word of the filler word sequence. In addition, filled pauses (FPs) and discourse markers (DMs) are distinguished because they are quite different phenomena. Therefore, the structural event types used for the boundary detection tasks are: the end of an SU, the edit IP, the end of a DM, and the end of an FP.

It needs to be decided whether to train one model that learns to distinguish between every event type, or separate models that only learn to distinguish among a subset of event types. Some of the events shown above can co-occur, for example, SU and the end of a filler, but some do not co-occur (e.g., SUs and IPs).

The approach we adopt for the baseline system is to train a separate prosody model and LM for each event, that is, a binary classification for SU versus non-SU, IP versus non-IP, DM_END[1] versus non-DM_END, and FP_END versus non-FP_END. Since the distributions of these events are quite different, we believe that combining them into a single model may degrade the quality of the decision trees by masking the characteristics of the minority classes. In addition, the possible co-occurrence of two events will increase the total number of unique classes and may fragment the training data. We expect the models will be better tailored to each task by implementing the structural event detection tasks as multiple classification tasks. One drawback of such an approach is that the model cannot exploit the fact that some events often co-occur or do not co-occur at all; however, a post-processing step that is applied presumably will be able to address some of the problems that result from the separate modeling approach.

[1]DM_END means the end of a discourse marker word sequence. Similarly FP_END means the end of a filled pause.

5.1.2 Training Procedures

Data Preparation

As described earlier, structural event detection tasks are evaluated on both human transcriptions and recognition output. Note that when applying the models that are trained from human transcriptions to recognition output, there is likely a mismatch between training and testing. Our preliminary experiments showed that a model trained from the reference transcriptions yielded better performance than one trained from the recognition output when evaluating on the recognition output testing condition. Therefore, we used the reference transcriptions for model training and apply the resulting models to both the reference transcription and recognition output test conditions.

Figure 5.1 shows a diagram of how training data is obtained for the LM and prosody model. Speech needs to be segmented into shorter waveforms for later processing. For BN, reference "segments" are provided, which were generated by the transcribers based on pause and syntactic information. For CTS, segmentation is done automatically. In the training data, each word in a transcription has associated beginning and ending time; therefore, the speech data can be segmented when the pause between two adjacent words is greater than a pre-defined threshold (length greater than 0.3 seconds for CTS).

Before forced alignment, a step called 'text normalization' is performed. The purpose of this procedure is to "normalize" the words to match the vocabulary of the speech recognizer. For example, different backchannel words such as *"uhhuh"* or *"umhmm"* are mapped to a single token *"uhhuh"*, and compound words are split to avoid out-of-vocabulary words. The normalized words together with the structural event annotation in the original transcriptions compose joint sequences of words and structural events, which are then used for the hidden event LM training.

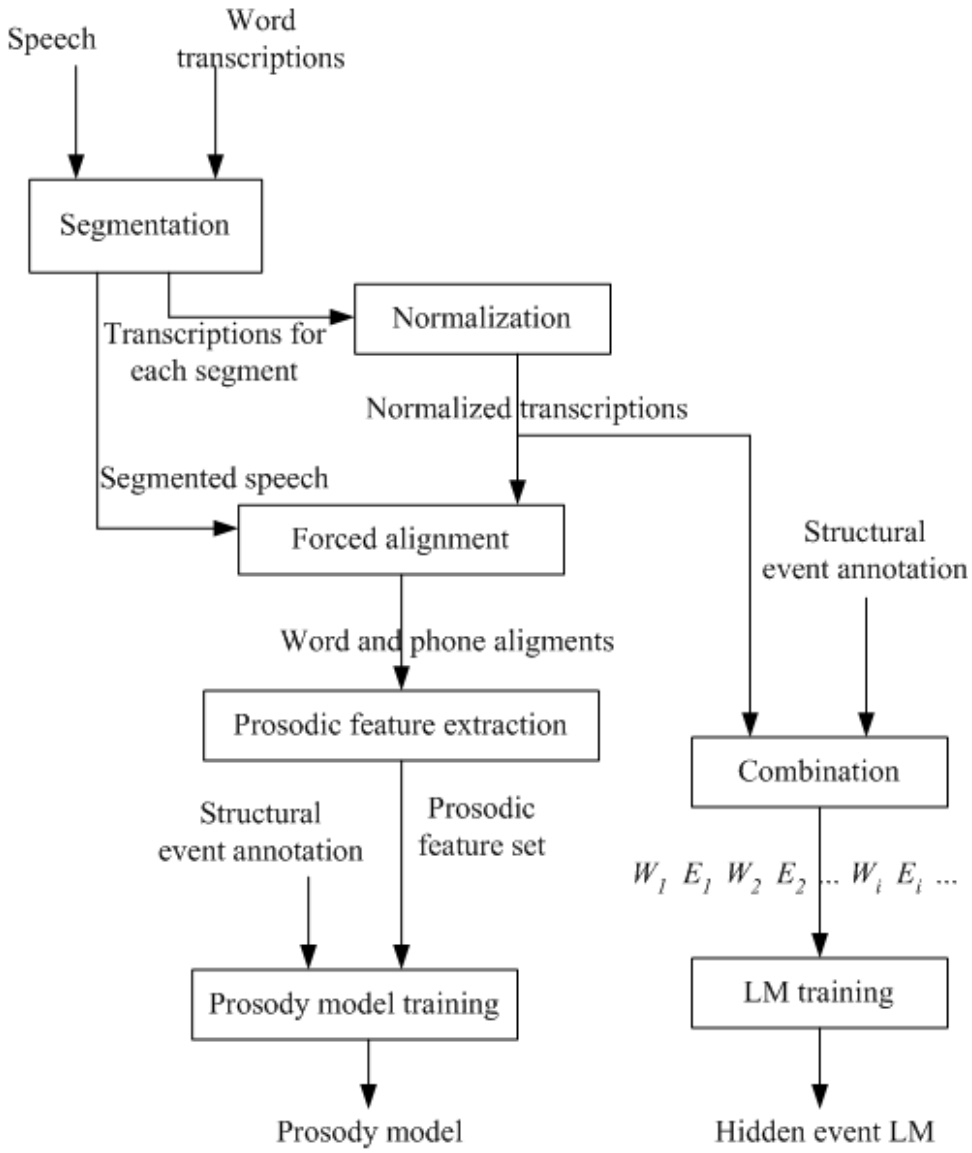

Fig. 5.1. Data preparation for model training.

The normalized transcriptions (without events) are then force aligned with the speech data to obtain the word and phone level alignments.[2] Forced alignments are obtained using SRI's large vocabulary speech recognizer [78]. The prosodic features are then computed using this alignment information as described in Chapter 4. Al-

[2]Even though word alignment information is provided in the training data, phone-level alignment is needed for prosodic feature computation; therefore, the re-alignment is conducted.

though prosodic features are extracted from the word or phone alignments, word identity information is not explicitly used in these features. Only phone identity is used to obtain normalized phone duration features. To compute F0 features, speaker information is needed. On CTS, it is straightforward to obtain this information, since each channel corresponds to one speaker. On BN, automatic clustering [87] is used to obtain a pseudo speaker label for each speech segment.[3] After prosodic feature computation, the data for the prosody model training is ready: a vector of prosodic features is associated with each word boundary, plus the structural event type corresponding to that boundary.

Model Training

As described in Chapter 4, a decision tree is used as the prosody model. Since the decision tree learning algorithm can be inductively biased toward the majority class (in the structural event detection case, non-event boundaries), the minority class may not be well modeled. Hence, for each two-way classification task in the baseline system, when training the decision tree using the prosodic features, we have randomly downsampled the training data in order to allow the decision trees to learn the inherent properties for the event classes. Downsampling evens out distributions by creating classes that each has the same number of tokens as the smallest minority class. Other ways to address the skewed class distribution problem will be investigated in Chapter 7.

From the joint word and event sequence (W, E), a 4-gram word-based hidden event LM is trained, with Kneser-Ney smoothing. Unlike prosody model training, for LM training, word sequence information must be preserved; therefore, no sampling is used.

[3] In the reference transcriptions, each segment has an associated speaker label. We chose not to use this information because in testing such information is unavailable; there would be a mismatch between training and testing if reference speaker information is used in training.

5.1.3 Testing Procedures

Testing Steps

Figure 5.2 shows the steps for testing the structural event detection models. These steps apply to either human transcriptions or recognition output. In both cases, word alignment information is available. When evaluating on the human transcriptions, the test data provided by NIST contains word alignment information. When evaluating on the recognition words, alignment information is available from the recognizer output. Each step in testing is briefly described below.

First the transcription with word alignment information is used to segment the speech (the whole conversation in CTS or a show in BN). This is done by finding a pause that is longer than a predefined threshold (0.5 second in our experiments) between two adjacent words.

For each segment of speech, a forced alignment is performed to its corresponding word transcription to obtain more detailed phone and word level alignments for later prosodic feature computation. In this step, a speaker label is also added to each segment of speech. For CTS, this is straightforward, since each channel corresponds to speech from one speaker. For BN, automatic speaker labeling is performed.[4] Prosodic features are then computed using the forced alignments and speaker information. Finally, the prosody model and the LM are combined in an HMM to obtain the final structural event hypotheses.

Evaluation Conditions

To achieve the best system performance, the prosody model and the LM are combined using an HMM. However, each model is also evaluated individually in order to understand their individual contributions. Hence, there are three evaluation conditions investigated:

[4]Different automatic speaker labeling methods will be discussed in Chapter 10.

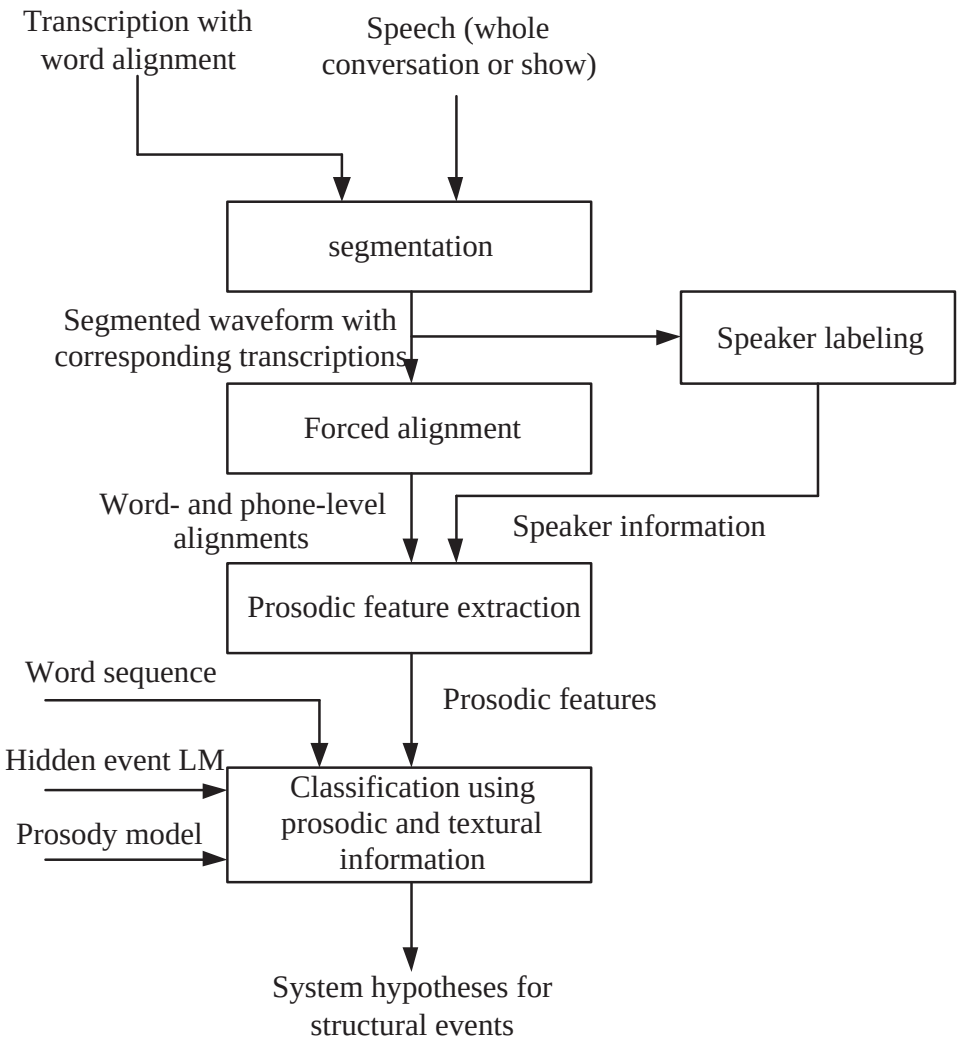

Fig. 5.2. System flow diagram of the testing procedure.

- Prosody model alone: The decision tree prosody model estimates the posterior probability of an event given the prosodic features at each word boundary. Since the decision trees are trained from a balanced training set, when testing the prosody model alone on the non-downsampled test data, the posterior probabilities generated by the decision trees are adjusted. Assume the prior probabilities of the two classes are $P(C_1)$ and $P(C_2)$ and the posterior prob-

abilities from the decision tree are $P_{DT}(C_1|F)$ and $P_{DT}(C_2|F)$, then for class C_i, the adjusted posterior probability is calculated as follows:

$$P_{adjusted}(C_i|F) = \frac{P_{DT}(C_i|F) \times P(C_i)}{P_{DT}(C_1|F) \times P(C_1) + P_{DT}(C_2|F) \times P(C_2)} \tag{5.1}$$

- LM alone: For each word boundary, the hidden event LM is used to compute the posterior probability of an event $P(E_i|W)$ using the forward-backward algorithm.

- Combination of the LM and the prosody model: The HMM is used to integrate the prosody model and the word-based hidden event LM as described in Chapter 4. When a downsampled balanced training set is used for prosody model training, then in the non-downsampled test set, the posterior probabilities are adjusted to account for the mismatch between decision tree training and testing as follows:

$$P_{DT_{adjusted}}(E_i|F_i, W_t) \propto P_{DT_{original}}(E_i|F_i, W_t) \times P(E_i) \tag{5.2}$$

  Substituting this into Equation (4.8), the most likely event sequence using the combined prosody model and LM can be obtained using the posterior probabilities directly from the decision tree:

$$\hat{E} = \arg\max_{E} P(W, E) \prod_{i} P_{DT_{original}}(E_i|F_i, W_t) \tag{5.3}$$

Posterior probabilities are generated for each word boundary in all these conditions: the LM alone, the prosody model alone, and the HMM combination approach. A threshold of 0.5 is used to generate the final decisions for all the binary structural event detection tasks, that is, the class whose posterior probability is greater than 0.5 is chosen. This is because our goal is to minimize the overall classification error rate, and the errors associated with each class (event vs. non-event) are assumed to be equally costly.

5.2 Baseline System Performance

The prosody model, LM, and their combination are evaluated for the structural event detection tasks, using human transcription or recognition output for the CTS and BN corpora. Experiments for each subtask are described in the following subsections.

5.2.1 Task 1: SU Detection

Setup

This is a two-way classification task, where non-SU boundaries are distinguished from SU boundaries. During the model training and testing, SU subtypes are not distinguished. Hence, the "SU" class includes statements, questions, backchannels, and incomplete utterances. All the other boundaries are grouped into one class "non-SU".

Training and test data are those used in the RT-03 evaluation, which is described in Chapter 3 (see Table 3.2). The decision tree prosody model is trained from a downsampled training set. Evaluation is conducted on both the human transcriptions (REF) and recognition outputs (STT), using the LM and the prosody model individually, as well as their combination. To obtain a baseline performance for the SU detection task, the NIST error rate is used, which allows comparisons with the other systems using the same data. Results are also reported using classification error rate (CER) to examine the boundary detection performance.

SU Detection Results for CTS

Table 5.1 shows the SU detection results for CTS. As shown in the table, the LM performs better than the prosody model alone, and the model combination outperforms either model alone. The word-based LM does not generalize well to unseen cases: it can only accurately detect SU boundaries when the word context

has occurred in the training data. On the other hand, combining with the prosody model is better at generalizing to unseen test conditions. This can be seen from Table 5.2, which shows the deletion and insertion error rate using the LM and the prosody alone and in their combination for the reference transcription condition. In the model combination case, there are fewer missed SU boundaries (deletion errors), although there is an increase in false alarms (insertion errors) compared to using the LM alone.

Table 5.1
CTS SU detection results using the NIST SU error rate (%) and the boundary-based CER (% in parentheses) on human transcriptions (REF) and recognition output (STT), for the LM and the prosody model individually, and in combination. The baseline error rate, assuming there is no SU boundary at each word boundary is 100% for the NIST SU error rate and 15.7% for CER.

CTS			
	LM	**Prosody**	**LM+Prosody**
REF	42.02 (6.56)	68.77 (10.73)	36.24 (5.65)
STT	53.25 (8.31)	70.98 (11.07)	46.52 (7.26)

Table 5.2
Deletion and insertion error rates (NIST SU error rate in %) for the CTS REF condition, using the LM and the prosody alone and in their combination.

CTS			
	DEL	INS	Total
LM	27.15	14.87	42.02
Prosody	62.58	6.18	68.77
LM+Prosody	18.86	17.38	36.24

The word errors in the recognition output have a negative impact on both the prosody model and LM, with the LM more severely affected (i.e., there is a greater relative SU detection error increase in the STT condition for the LM than for the prosody model). Since the LM is more dependent on lexical information than the prosody model, it follows that it would be less robust in the face of word errors. The prosody model is also indirectly impacted because the prosodic features are extracted from the word alignments using transcriptions containing word errors, so they should be less accurate than using human transcriptions. In addition, incorrect phones in the STT output affect prosodic feature extraction; for example, it impacts the normalization of phone duration, which uses phone identity.

An analysis of the decision tree created during training highlights what prosodic features are used most often for SU boundary detection. Table 5.3 reports the feature usage for this SU task. Feature usage reflects the percentage of times decisions involve a certain feature when classifying all the training samples [12]. Features that are used higher up in the decision tree have higher usage values. Only those features with the feature usage value greater than 4% are listed in the table. Descriptions of the feature abbreviations are given in Appendix B. As can be seen, duration is extremely important for SU detection on CTS data, including pause duration and phone level (e.g., vowel) duration. Another useful feature is whether there is a turn change, which in most cases implies that the speaker has finished his or her utterance, thus signaling an SU boundary.

SU Detection Results for BN

SU detection results using BN corpus are shown in Table 5.4. Similarly to the CTS results, the combination of the LM and the prosody model yields better performance than either model alone, and there is a performance degradation in face of speech recognition errors. However, the contribution from the prosody model to the combined performance for BN is greater than for CTS, i.e., there is about a 20%

Table 5.3
Feature usage (%) for SU detection on CTS.

Feature	**Feature Usage (%)**
PAU_DUR	23.613
WORD_DUR	20.146
PREV_PAU_DUR	12.647
MAX_VOWEL_DUR_NSP	11.430
TURN_TIME_N	7.963
TURN_F	4.382

error rate reduction after combining with the prosody model for BN, compared to around 13% for CTS on the reference condition. Table 5.5 shows the deletion and insertion errors for the reference condition. Similar to the CTS results, adding the prosody model yields fewer deletion errors and increases the insertion errors.

Table 5.4
BN SU detection results using the NIST SU error rate (%) and the CER (% in parentheses) using the prosody model, the LM, and their combination. Results are shown for both REF and STT conditions. The baseline error rate is 100% for the NIST SU error rate and 7.2% for CER.

BN			
	LM	**Prosody**	**LM+Prosody**
REF	80.44 (5.79)	85.67 (6.17)	64.75 (4.66)
STT	84.71 (6.10)	85.67 (6.17)	69.07 (4.97)

Recall from Table 3.2 that there is a smaller percentage of SUs in BN than CTS. Since the number of reference SUs is used as the denominator in the NIST error rate calculation, the same number of misclassified boundaries will result in a higher

Table 5.5
Deletion and insertion error rates (NIST SU error rate in %) for the BN REF condition, using the LM and the prosody model alone and in their combination.

BN			
	DEL	INS	Total
LM	66.00	14.44	80.44
Prosody	30.42	55.26	85.67
LM+Prosody	37.01	27.74	64.75

NIST error rate in BN than in CTS. This partly explains why the NIST SU error rate is generally higher on BN than on CTS. The boundary-based CER is lower for BN than for CTS, although the simple baseline performance is also lower (7.2%) for BN than for CTS (15.7%).

Comparing the SU detection results on BN and CTS (Table 5.1 and Table 5.4), we notice that the performance of the LM alone is worse on BN than on CTS. There are several reasons for this. First, the training data size is smaller for BN than for CTS. Second, these two corpora differ in speaking style. In conversational speech, there are many first person pronouns and backchannel words, which are very good signals for SU boundaries; whereas, for BN, sentence initial and final words are quite variable and thus the data is more sparse. These characteristics make the BN SU detection a harder task, which is reflected by the higher NIST SU error rate (which is normalized by the event priors).

As expected, the performance of the LM alone degrades more than that of the prosody model alone on the STT condition. The degradation (relative error rate increase for the LM alone) due to the STT errors is lower on BN than on CTS, which can be attributed to the better word recognition accuracies on BN than CTS. In contrast to CTS, we observe that the performance of the prosody model alone

does not degrade on the STT condition.[5] In fact, for the STT condition, the prosody model alone yields a performance similar to the LM alone.

Table 5.6 shows the most frequently used prosodic features by the decision tree for the BN SU detection task. These differ from those for CTS, although the pause duration at the word boundary is the most frequently used feature in both cases. For BN, pitch plays a more important role than for CTS; whereas, phone and word duration is more important for CTS. The pause duration before the current word is not as useful for BN as for CTS. These differences may be due to the fact that most of the speakers in BN are professional reporters reading tele-prompted text, and they use pitch change to reflect the sentence structure more consistently than people engaged in conversational speech. In addition, CTS involves conversational speech between two speakers, and thus the pause information for one speaker is affected by the speech activity of the other speaker.

Table 5.6
Feature usage (%) for SU detection on BN.

Feature	**Feature Usage (%)**
PAU_DUR	44.299
TURN_TIME_N	20.966
LAST_RHYME_DUR_PH_ND_bin	17.713
F0K_DIFF_LAST_KBASELN	4.539

Note that for the RT-03 structural event detection tasks, subtypes of SU are not evaluated; therefore, those results are not reported here. A preliminary investigation [88] indicates that different prosodic features may be important for identifying some SU subtypes (e.g., the rising pitch for a question SU). Additionally, the word-based

[5]The exact same SU error rate for the prosody model alone is just a coincidence; the errors are different.

N-gram LM is less effective at distinguishing questions from statements than for detecting backchannels.

5.2.2 Task 2: Filler Word Detection

Setup

Since filler words are infrequent phenomena in BN, we focus only on CTS for this task. Here we only detect filled pauses (FPs) and discourse markers (DMs) for filler word detection. Explicit editing terms are not considered due to the lack of a good modeling approach for them and their infrequent occurrence. We expect that FPs and DMs have sufficiently different characteristics that separate models are built for each. As explained earlier, a boundary detection approach is first used for determining the end boundary of the filler words. Both the filled pause and discourse marker detection tasks involve a 2-way classification. After detecting the end of a discourse marker phrase, we search backward over words that appear on a pre-defined discourse marker list to determine the onset of the discourse marker phrase. Filled pauses contain only one word; therefore, knowing the end of an FP is equivalent to detecting the FP.

Evaluation is performed using both the human transcription and recognition output from CTS. As for the SU detection task, the prosody model and the LM are evaluated individually and in combination.

CTS Filler Word Detection Results

The results for filler word detection on the CTS corpus are shown in the first two rows in Table 5.7. Again we observe a performance degradation due to the errors in the STT output. For tasks such as filler word detection, which strongly depend on word identity, inaccurate STT output severely affects the LM's performance. The increase of the filler error rate in the STT condition (compared to using human

transcriptions) is more dramatic than for the SU detection task. The combination of the LM and the prosody model is not substantially better than using the LM alone for filler detection, in contrast to what we have observed for SU detection on both CTS and BN.

Table 5.7
Results for CTS filler word (including FP and DM) detection, FP detection, and DM boundary detection using NIST error rate (%) and CER (% in parenthesis) for the prosody model, LM, and their combination. Results are for both the REF and STT conditions. The baseline CER is 8.3% for filler word detection, 3.6% for FP detection, and 2.8% for DM boundary detection.

CTS				
		LM	**Prosody**	**LM+Prosody**
Filler word	REF	21.18 (1.76)	63.02 (5.23)	20.78 (1.72)
	STT	49.22 (4.09)	79.59 (6.61)	48.10 (3.99)
FP	REF	6.77 (0.19)	5.0 (0.14)	3.04 (0.09)
	STT	50.88 (1.42)	49.36 (1.38)	48.87 (1.37)
DM boundary	REF	39.48 (1.42)	96.81 (3.49)	38.71 (1.39)
	STT	55.52 (2.0)	96.85 (3.49)	56.14 (2.02)

A detailed look at the performance for FP detection alone shows some interesting trends, as can be seen in the two rows of the FP detection results in Table 5.7. The prosody model alone yields very good accuracy for FP detection. It is quite surprising that prosody information can be utilized alone to so accurately detect filler words when human transcriptions are used. Additionally, on both the REF and STT conditions, we observe that the prosody model performs better than the LM alone. We believe that the decision tree has learned very word specific prosodic features (e.g., by learning specific duration and pitch features for filled pauses). The DM results in the last two rows of Table 5.7 are for DM boundary detection, not for

DM extent. This represents better the performance of our models, since the 2-way boundary detection model is used for DM detection. Unlike FP, the prosody model alone performs poorly for DM detection. In addition, on the STT condition, the combination of the LM and the prosody model is not better than the LM alone.

The feature usage of the prosody model for each of the FP and DM detection tasks is shown in Table 5.8. Features such as vowel duration and rhyme duration play an important role for filler word detection. These features are very different than those chosen for the SU detection task on both the CTS and BN data. For example, the most used feature for SU detection is the pause after a word; whereas, for filler, especially FP detection, word lengthening is more informative.

Filler detection will not be a major focus in this thesis. When using the speech recognition output, whether the recognizer outputs the correct filled pause word largely determines FP detection performance since it then only involves simple keyword identification. For DM detection, there is some ambiguity because some discourse marker words are confusable with words used in other situations. Higher level textual information together with word-dependent prosody models is needed for resolving this ambiguity.

5.2.3 Task 3: Edit Word and IP Detection

Since the edit word detection and IP[6] detection tasks are highly correlated, we describe the two tasks together in this section. Our focus is on CTS data because edit disfluencies are infrequent in BN.

Overview of Edit Detection

Edit word detection is an extent detection task; whereas, IP detection belongs to the boundary detection category. Since speakers are still fluent at the starting point

[6]An IP here means the IP inside an edit disfluency, not including the point before the filler words, which are not the editing terms in an edit disfluency.

Table 5.8
Feature usage (%) for the FP and DM detection tasks in CTS.

Task	**Feature Usage (%)**	
FP	MAX_PHONE_DUR_N	14.869
	MAX_PHONE_DUR_Z	14.325
	ENERGY_WIN_DIFF_HIHI_N	13.173
	LAST_VOW_DUR_N_bin	11.741
	LAST_RHYME_DUR_PH_ND_bin	8.079
	LAST_VOW_DUR_Z_bin	5.403
	AVG_PHONE_DUR_Z	5.141
	STR_RHYME_DUR_PH_bin	4.963
	AVG_PHONE_DUR_ZSP	4.501
	PREV_PAU_DUR	4.162
	F0K_WRD_DIFF_LOLO_N	4.073
DM	WORD_DUR	19.338
	LAST_RHYME_DUR_PH_bin	15.178
	TURN_TIME	12.832
	AVG_VOWEL_DUR_Z	5.219
	AVG_VOWEL_DUR_N	4.570

of an edit disfluency, it is likely that there are no acoustic or lexical cues at that location, but there might be some cues at the point when the speaker interrupts his or her speech. Therefore, our baseline approach is to use the prosodic and lexical features to detect the interruption point (IP) and then use knowledge-based rules to locate the corresponding reparandum onset for the hypothesized IPs. Figure 5.3 shows a system diagram for this method. The final system output (for both IP and edit word detection) is from the combination of the HMM-based IP detection, rule-

based reparandum onset detection, and repetition pattern detection. Each box in the figure is described below.

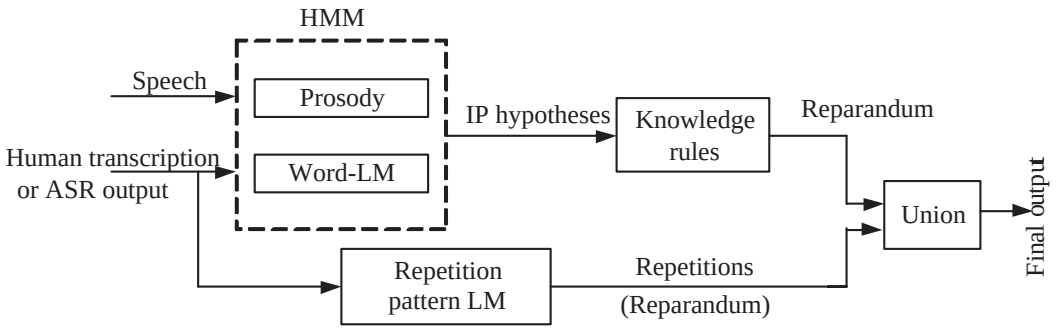

Fig. 5.3. System diagram for edit word and IP detection.

HMM-based IP Detection

The top left box in Figure 5.3, shown with a dashed line, represents a two-way classification model used to determine whether or not there is an edit IP at an interword boundary position. A decision tree prosody model is trained from a down-sampled training set. A hidden event LM is trained to model the joint distribution of words and IP events (e.g., *"I ⟨IP⟩ I go to school"*). These two models are then combined using an HMM. The final output from this module is the IP hypothesis for each word boundary.

Modeling Repetition Patterns

A word-based N-gram LM can only learn certain frequently occurring disfluencies from the training data and cannot generalize to other related disfluencies using different words. For example, in *"I hope to have to have lots of dinner parties"* (with *"to have"* repeated), a regular word-based hidden event LM would fail to detect the

IP in this utterance.[7] Such a failure would also affect the speech recognition task for which the purpose of the LM is to calculate the probability of word strings. To address this issue, the word-based LM has been modified to account for repetitions. Currently only repetitions are handled because they are the most constrained and frequently occurring edit disfluencies in CTS (more than half of all edit disfluencies are repetitions).

To train a LM that can deal with repetition patterns, the training corpus is processed in the following way. For each repetition in the training data, we remove the reparandum region to obtain a cleaned-up utterance, and record the repetition pattern. For the example above, the cleaned-up text is *"I hope to have lots of dinner parties"*, and the repetition is mapped as follows:

		to		have		to		have	
START		ORIG-1		IP		REP-1			END

The pattern sequence in the example is thus 'START ORIG-1 IP REP-1 END'. The number after '-' in the pattern denotes the position of that event in either the reparandum or the repeat region. An N-gram LM is trained from the counts that are obtained from both the cleaned-up text and the counts of such repetition patterns. Note that the whole repetition pattern sequence is modeled as shown in the pattern example (e.g., IP as well as the reparandum onset); whereas, the hidden event LM captures only the hidden event IP information.

During testing, for each word boundary, repetition events are hypothesized based on the valid state transitions and whether a word matches a previous word. Each hidden event in a repetition pattern has properties representing where it occurs in the pattern, from which a possible valid next event can be inferred. During trellis decoding, only valid state transitions are considered. Figure 5.4 shows the state transitions for repetitions with up to three repeated words. This LM can also be interpreted as a class-based LM, i.e., when a word sequence is found to be repeated, the words in the reparandum region are mapped to the class tags (i.e., the tokens used

[7]The word-based LM would fail to detect the repetition in the utterance because "*to have to have*" was never observed in the training data.

in the repetition patterns). In addition, we assume there is not any lexical context associated with these repetition patterns in the N-gram LM, which is equivalent to allowing such a pattern to occur for any word choice. The probability of a word sequence is calculated in the same way as in a word-based N-gram LM for fluent words up until the 'IP' point, then in the repetition, the pattern N-gram probability is used instead of the word-based probability. An advantage of this approach is that it can detect more than the frequently occurring repetitions in the training data. Note that in the repetition detection model, there are some cue words that are not allowed to be mapped to the repetition pattern (e.g., *"yeah", "uhhuh", "right"*), since the repetitions of these words generally are not edit disfluencies.

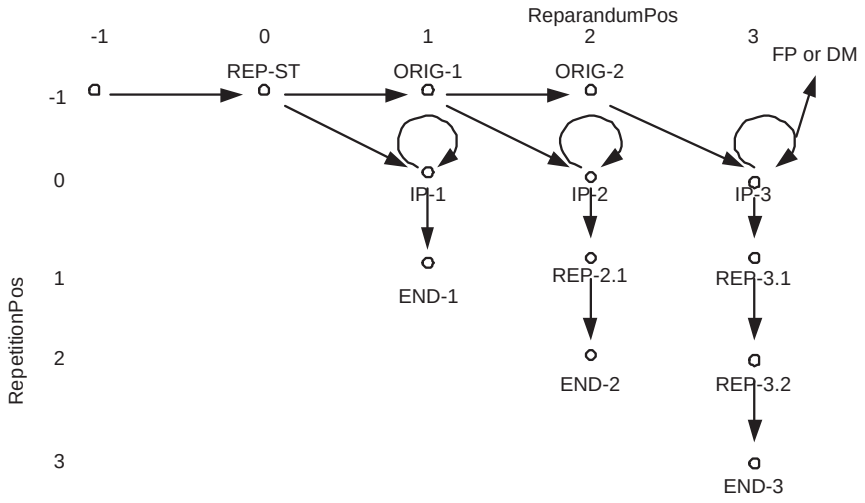

Fig. 5.4. Valid state transitions for repetitions of up to 3 words. The X and Y axes represent the position in the reparandum and repetition regions respectively, with events denoted as ORIG- and REP-. In ORIG-n, n means the position of a word in the reparandum; in REP-m.n, m is the total number of repeated words and n represents the position of the event in the repeat region. Optional filler words are allowed after the IP in the transition.

Rule-based Reparandum Detection

The rule-based knowledge box in Figure 5.3 applies heuristic knowledge to determine the extent of the reparandum region of a disfluency after the interruption point is detected. Linguistic investigations (e.g., [42]) suggest that people tend to start from the beginning of a constituent in repetitions or revisions (e.g., repeating the function words). For example, a revision disfluency may be: *"a red a blue car"*, where the speaker starts from *"a"* when trying to correct the word *"red"*. In this example, if the IP is correctly hypothesized at the interword boundary, that is *"a red* ⟨*IP*⟩ *a blue car"*, then we can go backward to find whether the same word as the word after the IP (*"a"* in this example) has occurred before the IP and thus determine the onset of an edit disfluency. For a word boundary that has a system IP hypothesis from the HMM component or that follows a word fragment, Figure 5.5 shows how rule-based knowledge is applied to determine the reparandum starting point. During the post-processing of IPs, the system generated SU information is also used. In the bottom right box in Figure 5.3, when looking forward to search for a word that matches the word after the IP, the search stops when it hits a system SU hypothesis or a cue word (e.g., "*and*", "*but*") to avoid false alarms.

Edit and IP Detection Results

Table 5.9 shows the results for edit word and IP detection using the CTS corpus. When testing on the human transcriptions, one important type of information is the occurrence of word fragments, which often signals a disfluency interruption point. For the human transcription results shown in Table 5.9, this information is used, because our goal is essentially to recognize fragments from the speech signal. However, most currently available speech recognizers do not hypothesize the occurrence of word fragments, so there is a difference between the REF and STT conditions related to the availability of word fragment information.

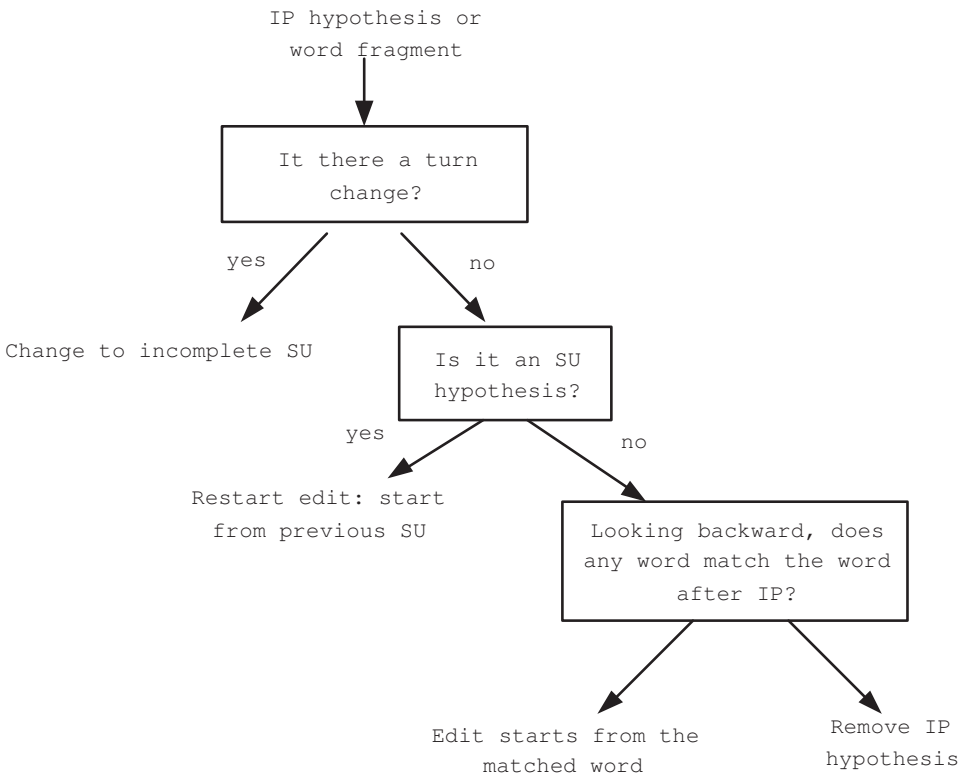

Fig. 5.5. A rule-based method for determining the reparandum region after IPs are hypothesized. SU hypotheses are used in the rules.

As can be seen from Table 5.9, when the prosody model is used alone, none of the edit IPs and thus none of the edit words are detected. The NIST error rate is 100%, which are all deletion errors. Note that when the prosody model is used alone, we do not have access to the word repetition module (which uses textual information). Experiments in [18] have shown that useful prosodic cues exist at the interruption points, but the performance of the prosody model was not investigated on non-downsampled test data in that study. In prior work, we have found that on a downsampled test set the prosody model alone yields a better accuracy than chance performance of 50% for IP detection [88]; however, on the non-downsampled data, Table 5.9 shows that the prosody model alone yields only chance performance. This may be because the posterior probability generated by the prosody model is

Table 5.9
CTS edit word and IP detection results using NIST error rate (%) and CER (% in parenthesis) for the prosody model, the LM, and their combination. Results are for the REF and STT conditions. The baseline CER is 8.3% for edit word detection, and 4.8% for edit IP detection.

CTS				
		LM	**Prosody Only**	**LM+Prosody**
Edit word	REF	59.14 (4.91)	100 (8.3)	59.06 (4.90)
	STT	88.02 (7.31)	100 (8.3)	87.86 (7.29)
Edit IP	RER	41.1 (1.96)	100 (4.78)	40.47 (1.93)
	STT	81.6 (3.9)	100 (4.78)	81.26 (3.88)

insufficiently high to overtake the low prior probability of an 'IP' event. As shown in Equation (5.2), when the prosody model is used alone, the final posterior estimation at a word boundary is obtained by adjusting the decision tree's output using the prior probability information. The prior probability for an IP is about 4.5% in CTS (as shown in Table 3.2); therefore, if the decision tree's posterior probability output for the IP event is not greater than 0.955, then the non-IP event will have a higher adjusted posterior probability at that boundary. This is in contrast to FP detection (see Table 5.7) on non-downsampled data, where the prosody model performs very well even though the FP event has a lower prior probability (2.9%) than for the IP event (4.5%). The effectiveness of the FP prosody model is attributed to the reliability of the prosodic features for that task, i.e., the prosody model has a very high posterior probability for some FP test samples (almost 1). In contrast, the prosody model for the IP detection task is less reliable given the current set of prosodic features used in the prosody model.

Results in Table 5.9 indicate that the LM performs significantly better than chance; however, word errors significantly affect the robustness of the LM because it

relies heavily on the word identities and patterns. An analysis of the results shows that most of the IPs correctly detected by the hidden event LM are repetitions. This makes sense since repetitions are common in the training set. Additionally, it is difficult to capture the properties of revisions and restarts using a simple word-based N-gram model. Even though the prosody model only achieves chance performance when used alone on the non-downsampled test set for IP detection since the posterior probabilities do not win over the high priors of the non-IP event, we find that the combination of the LM and prosody model slightly outperforms the LM alone.

On downsampled data, the decision tree learns some important prosodic features for IP detection. The feature usage of the prosody model trained from a downsampled training set is shown in Table 5.10.

Table 5.10
Feature usage (%) for IP detection on CTS corpus.

Feature	**Feature Usage (%)**
PAU_DUR	19.933
AVG_PHONE_DUR_N	17.624
TURN_TIME_N	10.984
WORD_DUR	10.213
SLOPE_DIFF	6.683
AVG_PHONE_DUR_ZSP	5.412
AVG_PHONE_DUR_NSP	4.653

5.2.4 Summary for All the Tasks

Table 5.11 summarizes our baseline system performance in NIST error metric for all the structural event detection tasks on the human transcription and recognition

output for the CTS and BN corpora. Since these tasks are benchmark tests, using the NIST standard metric enables comparison with other systems.

Table 5.11
System performance (NIST error rate in %) for all the structural event detection tasks on CTS and BN test sets. Results are presented for both the REF and STT conditions.

	BN		**CTS**	
	REF	**STT**	**REF**	**STT**
SU	64.75	69.07	36.24	46.52
Edit word	51.37	100.39	59.22	87.99
Filler word	9.22	52.45	18.07	47.97
IP	17.51	74.47	27.13	65.75

Since separate classifiers are used for each task, there can be conflicts between different classifiers' decisions at an interword boundary. SU and edit IP conflicts are reconciled by looking at the posterior probability of the SU hypothesis since that value is more accurate than the IP posterior probability. When the SU posterior is higher than a predefined threshold, the SU hypothesis is preserved; otherwise the IP or edit hypothesis is used. For example, *"That is great. That is great."* If the repetition detection hypothesizes it as a repeat (edit disfluency) and the SU detection model hypothesizes this as a potential SU, then we check the posterior probability of the SU result. In this case, if it is higher than 0.85, then the edit word and IP hypotheses are removed and the SU hypothesis is preserved.

As shown in Table 5.11, performance degrades dramatically on the STT condition for all the tasks. Notice that the degradation on the SU detection task is less than for the other tasks. Some possible reasons for its greater robustness are: the prosody model for SU provides more help and is relatively more robust to recognition errors than the word-based model, the SU LM is not dependent on a list of the key words (as in the case of filler words detection) or specific patterns (as in the case of edit

word detection), and there are more SU events in the training data and so the model is better trained. Additionally, some of the degradation for edit disfluency and IP detection is due to the fact that in the reference transcription word fragments are provided, which always signal IPs; whereas, in recognition output, this information is unavailable. We also find greater degradation on the filler and edit detection tasks on the STT condition for BN than for CTS. This is probably because these events are more undertrained and a larger percentage of edit disfluencies contain word fragments on BN.

Interestingly, we observe from the table that the edit word detection error rate on BN is not worse than CTS for the reference condition, even though the percentage of edit words is much smaller on BN than CTS, which significantly affects the denominator used in the performance measure. This suggests that to some extent edit word detection is a relatively easier task on BN than CTS, which makes sense because of the different speaking style of the two corpora. Many edit disfluencies on BN are repeats and simple revisions, some of which are due to reading errors.

5.3 Chapter Summary

In this chapter, we have described our structural event detection baseline system performance. An HMM incorporating word information and prosodic features is used for each boundary detection task. Additional rules are used for further extent detection. We have observed effective acoustic-prosodic cues at the event boundaries and these features vary for different tasks and corpora. For example, a pause after a word is an important feature for SU boundaries, word lengthening is more important for filler words, and pitch was found to be more effective for BN SU detection than for CTS. The LM alone generally outperforms the prosody model, and the combination of the two models usually performs better than either individual model. Comparisons of performance on the REF and STT conditions suggest that the incorrect words contained in the recognition output are more detrimental to the LM than to the

prosody model. This makes sense since the former is more dependent on correct word identity.

In summary, the combination of multiple knowledge sources improves performance for the structural event detection tasks. Different models and knowledge sources have their own strengths in modeling different components in the structural event detection tasks. How to more effectively model each knowledge source and integrate them for improved system performance is a major focus for the remaining chapters in this thesis.

6. INCORPORATING TEXTUAL KNOWLEDGE SOURCES INTO THE HMM SYSTEM

In speech recognition, the role of a LM is to predict the next word based on its context or to disambiguate word candidates that are acoustically confusable. For such a purpose, a word-based N-gram LM performs very well, although research has also shown that incorporating more knowledge further improves performance [89,90]. For structural event detection tasks, it is likely that syntactic information will be even more important for determining SUs and disfluencies than for word prediction. In this chapter, we investigate whether textual information beyond words is helpful for detecting structural events in speech.

Another motivation for our investigation of using additional information beyond the word-level is the data sparsity issue in training the hidden event N-gram LM. A hidden event LM requires the use of an annotated training set, which typically requires more effort to produce than transcriptions. Therefore the data set available for training the hidden event LM is generally much smaller than the size of the corpus used to train a word-based LM used in speech recognition. A traditional class-based LM[1] is less affected by the sparse data problem and thus may generalize better since it groups similar words into classes, which leads to a decrease in the vocabulary size. In addition, we will also investigate using additional 'annotated' text material that does not accurately match the type of data we will test on.

This chapter is organized as follows. Section 6.1 reviews related LM techniques. Section 6.2 describes additional knowledge sources that we investigate, including automatically induced classes, part-of-speech tags, syntactic chunks, and other textual

[1]A class-based LM is in contrast to a word-based LM; the tokens in a class-based LM are the 'classes' of words.

material that imperfectly matches the test condition. Section 6.3 describes how these textual knowledge sources are combined together, as well as with the prosody model. Section 6.4 shows the experimental results for the SU detection task in the HMM system. Section 6.5 summarizes this chapter.

6.1 Review of Related Language Model Techniques

In our baseline system, a word-based hidden event LM is used. Before this LM is extended, we first review some related efforts aimed at improving LMs, most of which were developed in the context of speech recognition. We will review word-based LMs and their parameter estimation, word+class-based LMs,[2] and LM adaptation. Some of the techniques presented here will be applied to the structural event detection tasks.

- **Word-based LMs**

  The goal of a LM is to estimate the prior probabilities of a word sequence:

  $$P(W_1^n = w_1 w_2 ... w_n) = P(w_1)P(w_2|w_1)...P(w_i|H(w_i))... \tag{6.1}$$

  where $H(w_i)$ is the word history for a word w_i. If the history mapping function $H(w_i)$ contains the previous $n-1$ words of w_i, then it becomes an N-gram LM, the most widely used LM. Even though this LM only captures limited local information, it is very powerful in applications such as speech recognition and machine translation, where the training corpora are reasonably large.

  For parameter estimation, the maximum likelihood (ML) method is generally used to obtain estimates: $P(w|h) = C(w,h)/C(h)$. In ML estimation, if an event (w,h) does not occur in the training data, then it will be assigned a zero probability and thus the word sequence will have a zero probability. This problem is known as the sparse data problem. In order to address this problem, a

[2]These are sometimes called class-based LMs, too. We use 'word+class' to denote that both word and class information is used in this LM, in order to distinguish it from the POS- or chunk-based LMs described later that do not use word information.

variety of smoothing methods have been proposed [91–94]. These techniques are central to improving a LM's performance (such as in speech recognition). Most of the smoothing methods discount probabilities of events that have occurred in the training set, and use these probabilities to provide probabilities for the unseen events. Commonly used smoothing techniques include linear interpolation, which combines the higher order N-gram LMs with the lower order ones, and backoff for which a higher order LM is used when the frequency is greater than some threshold; otherwise, there is a back-off to a lower order LM. Note that the sparse data problem is not limited to word-based LMs. Smoothing methods are applicable to, for example, class-based LMs that are discussed next.

- **Word+class-based LMs**

  Class-based LMs are important since they reduce the number of parameters, addressing at least in part the data sparsity problem, and they have the potential to increase the generality of the LMs to unseen events. There are two ways to obtain the classes: either the classes can be automatically induced based on some statistical information in the training set, or existing linguistic classes such as POS tags or semantic tags can be used.

  There are two modeling methods that have been used for jointly utilizing word and class information for word prediction. The first is the conditional probabilistic model that calculates $p(w|h)$ in the following way [95]:

$$P(w_i|H(w_i)) = \sum_{C(w_i)} p(w_i|C(w_i)) \times p(C(w_i)|H(w_i)) \tag{6.2}$$

  where $C(w_i)$ is the mapped class for word w_i, also simplified as C_i. If a trigram LM is used and the word history is also mapped to their classes, then the formula above becomes:

$$P(w_i|w_{i-1}, w_{i-2})) = \sum_{C_i, C_{i-1}, C_{i-2}} p(w_i|C_i) \times p(C_i|C_{i-1}, C_{i-2}) \tag{6.3}$$

The second method uses a joint model, which calculates the probability of the joint word and class sequence as follows:

$$p(W,C) = \prod_i p(w_i, c_i | w_1^{i-1}, c_1^{i-1}) = \prod_i p(c_i | w_1^{i-1}, c_1^{i-1}) p(w_i | w_1^{i-1}, c_1^i) \quad (6.4)$$

And the probability for the word sequence W is obtained by summing up over all the class sequences:

$$P(W) = \sum_{c_i, c_2, \ldots c_n} P(W_1^n, C_1^n) \quad (6.5)$$

Heeman [96] reported perplexity reduction using the joint model of words and classes compared to a word-based trigram LM. However, using the conditional model as shown in Equation (6.2) has not yielded significant reductions in perplexity or speech recognition error rate [97]. The joint modeling approach appears to provide a superior performance over the conditional model [98], largely due to the importance of lexical information, which is lost in the conditional modeling approach.

In addition to the POS tags and the automatically induced classes, recently LM researches are attempting to exploit more syntactic structure information for word prediction. Intuitively people do not simply employ local word-occurrence information for word prediction; they use much higher level knowledge, such as syntactic, semantic, and pragmatic information, in the face of acoustic ambiguity. LMs have been proposed that are based on various grammar. For example, Wang et al [89, 99, 100] and Chelba [90] have shown significant word error rate reductions in large vocabulary speech recognition tasks by incorporating syntactic information.

- **LM Adaptation**

  Another theme of research for LMs is adaptation. As we have already seen, a large training corpus is needed to obtain reliable parameter estimation for the N-gram tuples. However, in a new application, it is likely that the existing corpus is different from the application under study in terms of genre,

vocabulary, etc. In this case, adaptation methods can be used to adapt LM parameters from the general domain to the new domain, usually making use of a limited size training corpus that matches the application. Another approach is to build topic-dependent LMs, and then in a new application, detect the topic in order to choose a LM with the matched topic or use or a mixture of LMs with weights optimized for the testing corpus.

We have only reviewed some LM research that is more related to the methods that are going to be used for structural event detection. There is much work on LMs, such as using latent semantic analysis [101], exponential LMs [102, 103], and skipping LMs [104]. We choose to not elaborate on them since they are less relevant to our work.

6.2 Various Knowledge Sources

The work described in the previous section has been used in speech recognition; however, our research focuses on using LMs to detect structural events in speech. The question is, how well do the insights gained from word prediction applications transfer to the detection of structural events. In this section, we introduce various knowledge sources that are used to expand our word-based hidden event LM. The impact of these sources is evaluated in Section 6.4.

Note that for word prediction in speech recognition, a joint model of words and classes provides richer information, and thus helps resolve ambiguity and yields better estimation of the following word given word history. For the structural event detection task, it is unclear that a joint model of words, classes, and events $P(W, C, E)$ will be effective, since the goal is different from word prediction, i.e., detecting the hidden events given word-related contextual information. Additionally since we have such a limited training set, and a joint model $P(W, C, E)$ contains more parameters and requires more training data, we choose not to adopt the joint modeling approach for structural event detection tasks. Rather each class-based LM is a separate hid-

den event LM, which is used to model the joint distribution of classes and structural events $P(C, E)$, where C is the class sequence corresponding to the word sequence W. Class-based LMs have the advantage of reducing the vocabulary size and thus address data sparsity to some extent, while potentially capturing semantic or syntactic similarity information. Various LMs are then loosely combined. The next subsections describe each of the combined LMs.

6.2.1 Word-LM

This model was used in our HMM baseline system in Chapter 5. The hidden event word LM models the joint distribution of word and event sequence:

$$\begin{aligned} P(W,E) &= P(w_1, w_2, ...w_i, e_i, ...) \\ &= P(w_1)p(e_1|w_1)...P(w_i|w_{i-3}, w_{i-2}, w_{i-1})P(e_i|w_{i-2}, w_{i-1}, w_i)... \end{aligned} \quad (6.6)$$

Note again that only the structural events (e.g., e_i=SU) are explicitly included during training, but not the 'non-event' token. In the equation above, there is no other event other than e_i; therefore, the previous 3 words are used as the word history for w_i in a 4-gram LM. The Kneser-Ney smoothing method is used to smooth parameter estimates.

6.2.2 Automatically Induced Classes (AIC)

The word classes are induced from the distributional statistics (i.e., the bigram counts) in the annotated training data, using the criterion of minimizing perplexity of the class-based N-gram. Incremental greedy merging is performed, starting with one class each for the M most frequent words, and then adding one word at a time [105]. This class induction algorithm generates a one-to-one mapping, that is, each word is mapped to one and only one class. The resulting LM can be applied efficiently since the right context is not needed to map the current word to its corresponding class.

For the structural event detection tasks, since events are valid tokens in the vocabulary, one question is how to deal with the structural events during class induction. As mentioned earlier, the 'non-event' is not explicitly included in the hidden event LM in order to avoid fragmenting the word context. However, when inducing classes, it is unclear whether this token should be preserved or not in the bigram statistics. One hypothesis is that it may capture information that similar words co-occur with this 'non-event' token, suggesting these words have similar roles in terms of structural event detection. Preliminary experiments have been conduced comparing the methods for automatic class induction: with and without adding the 'non-event' token explicitly to the word sequence, but there was no improvement in performance from including it. Therefore this 'non-event' token is omitted from class induction.

Table 6.1 depicts examples of some words and the resulting classes that were induced for the SU detection task in CTS. We observe similar words are sometimes grouped into one class, such as class-1, which contains words that are good signals of SU boundaries. Class-2 also contains many similar words, for example *"you're", "we're"*; however, there exist other words (e.g., *"fierce", "historically"*) that seem dissimilar to the words in this class. The criterion for class induction is to reduce perplexity, but it is possible that there is a mismatch between this criterion and the task of event detection.

6.2.3 Part-of-speech (POS) Tags

The automatically induced classes are derived in a data-driven way. These classes may have the advantage of representing word usage in the data set; however, they do not necessarily result in groupings with a clearly interpretable linguistic meaning. Additionally, there may be a mismatch between training and testing corpora, and thus the bigram statistics used to induce the classes may not match the test conditions well. In contrast, part-of-speech tags represent syntactic word class information, and are related to how the language is generated. Example POS tags

Table 6.1
Two examples of automatically induced classes for the CTS SU detection task, depicting member words and each word's probability given the class.

CLASS	$p(w\|c)$	**Word**
CLASS-1	0.577331	yeah
	0.355084	uhhuh
	0.0298036	huh
	0.0118913	wow
	0.00617145	bye-bye
	0.0106871	hum
	0.00903138	yep
CLASS-2	0.252747	you're
	0.252015	we're
	0.442491	they're
	0.0271062	who's
	0.0010989	fierce
	0.00769231	somebody's
	0.00879121	everybody's
	0.0032967	something's
	0.0021978	nobody's
	0.0014652	historically
	0.0010989	peripheral

include NN (noun), VB (verb), and WP (wh-pronoun). POS tags are investigated for structural event detection tasks. The goal is to model syntactically generalized patterns, such as the tendency to repeat prepositions and the type of words that tend to begin utterances. The hidden event LM is used to model the joint distribu-

tion of POS tags and structural events $P(P, E)$, where P is the POS tag sequence corresponding to the word sequence W.

POS tags are obtained via a statistical TnT tagger, which uses a trigram HMM-based approach [106]. For this method, POS tags are represented as states in the HMM, and the transition probabilities are the maximum likelihood probability estimation derived from the training data. Viterbi and beam search are used during decoding (tagging) to speed up processing. The POS taggers are trained for CTS and BN corpora respectively in the follow way.

- CTS training. The TnT POS tagger is trained using the Switchboard Treebank data [107] and then applied it to tag the training and testing data used for the structural event detection task. Similarly to [83], the identity of some cue words (e.g., backchannels, filled pauses) are maintained.

- BN training. For BN, there is no BN Treebank data that is annotated with POS tags. We started with WSJ Treebank, from which an initial tagger was trained. We then tagged the BN text corpus (which is used to train a word-based LM for speech recognition) and re-trained a final POS tagger from the tagged BN data. Although the data set used to train the tagger is inaccurate, we expect the large set of the training data may compensate for this to some extent. The retrained tagger is used to tag the BN training and testing sets for the structural event detection task.

6.2.4 Syntactic Chunk Tags

We have introduced simple word-based and class-based hidden event LMs (automatically induced classes and POS tags) that do not directly model syntactic structure. Notice that a POS-based LM captures some syntactic knowledge by modeling the co-occurrence of syntactic POS tags, but it does not directly represent the syntactic structure of the input utterance. In order to investigate whether modeling syntactic structure helps structural event detection, we investigate the use of chunk

parsing. A chunker is chosen rather than a full parser for several reasons. First, a full parser often requires a sentence as input, which is not available in our situation. Second, because spontaneous speech often contains ungrammatical and/or incomplete utterances, a full parser may either fail for the ill-formed utterances or provide misleading structures; whereas, a chunk parser tends to be more robust. Finally, a chunk parser requires less computational effort than a full parser.

Text chunks represent the non-overlapping phrases of a sentence and provide a rudimentary syntactic structure. Each word belongs to a phrasal constituent. Examples of text chunks are the beginning of a noun phrase (B-NP), inside a noun phrase (I-NP), the beginning of a verb phrase (B-VP), and outside a phrase (O). The example in Table 6.2 shows a sentence from the BN corpus with the associated POS and chunk tags. The simple structure of this sentence is shown below, in which each chunk is indicated by a square bracket with the chunk types following the left bracket '['.

> [**NP** the top selling car] [**PP** of] [**NP** nineteen ninety-seven] [**VP** was announced] [**NP** today] [**O** and] [**NP** the winner] [**VP** is] [**NP** toyota camry]

Text chunking is represented as a classification problem, for which a supervised transformation-based learning (TBL) approach [108] is used. Starting with a set of predefined rule templates, TBL learns an ordered list of rules in a greedy way. At each iteration, a new rule is generated (based on the rule templates) that corrects at least one error in the training set, and the best rule which has the highest score is preserved. A typical objective function in TBL is to optimize the evaluation function, i.e., the difference between the number of the positive and the negative examples when a rule is applied.

We chose to investigate the use of chunk information only on BN since sentences are more similar to written text and thus they may have a more constrained syntactic structure. Also since POS tags are the input to a chunker and we expect a higher

Table 6.2
The POS and chunk tags for a sentence from the BN corpus, *"the top selling car of nineteen ninety-seven was announced today and the winner is toyota camry"*.

Word	**POS tags**	**Chunk tags**
the	DT	B-NP
top	JJ	I-NP
selling	NN	I-NP
car	NN	I-NP
of	IN	B-PP
nineteen	JJ	B-NP
ninety-seven	NN	I-NP
was	VBD	B-VP
announced	VBN	I-VP
today	NN	B-NP
and	CC	O
the	DT	B-NP
winner	NN	I-NP
is	VBZ	B-VP
toyota	NN	B-NP
camry	NN	I-NP

tagging accuracy on BN than CTS, one might expect a better chunking accuracy on BN. We used those rules that are provided in the fnTBL program [108] and that were trained from the sections 1-21 of the Wall Street Journal part of the Penn Treebank. Such a text chunker is used to obtain the chunk tags for both the training and test corpus in the BN SU task. Similar to POS and automatically induced classes, a

hidden event LM is built from the joint chunk and event sequence (CH, E), where CH is the chunk tag sequence corresponding to a word sequence W.

6.2.5 Word LMs from Additional Corpora

Since the hidden event word LMs are trained using a small corpus annotated by LDC for the structural event detection tasks, it is likely that LMs trained from this data will be undertrained. Hence an important question arises: can LMs from other corpora annotated with structural events help to improve performance? We expect an additional corpus can help address the sparse data problem encountered when a word-based LM is trained using only the annotated corpus by LDC. We investigate using other corpora for the CTS and BN SU detection task.

- CTS. The LDC Treebank for the Switchboard data contains punctuation and disfluency annotations [19]. The annotation guideline differs from that used in the EARS program [71], such as the definition of "sentences" and a different interpretation of incomplete SUs versus restart edit disfluencies. This corpus contains about 1.4 million words, which is larger than the RT-03 CTS training set (about 480K words as shown in Table 3.2). The percentage of SU boundaries is about 12% of all word boundaries, which is similar to that in the RT-03 data.

- BN. For BN, there exists a large text corpus that is used to train the LM for BN speech recognition. The punctuation information in this corpus is highly similar to the SUs in the structural event annotation. This corpus contains about 130 million words, compared to 178K words available in the RT-03 BN data (see Table 3.2).

For these additional materials, we can either merge them with the annotated RT-03 training set, then train a hidden event word-based LM using the merged data, or we can train separate LMs from each and then combine the models. Since the

additional text corpora differ from the RT-03 training and testing data on annotation scheme, it is likely that building separate models and then combining them with different weights is the better option.

Note that unlike BN, for CTS only the LDC Switchboard treebank data is used. Presumably for CTS we could also utilize the text corpus that is used to train the LM for speech recognition by using punctuation to approximate SU events; however, our preliminary experiments have shown that such a LM does not yield any gain. We hypothesize this is probably because the punctuation in the large data set is not as carefully annotated for SUs as in the RT-03 CTS data. Additionally the sparse data problem is not as serious in CTS as in BN (e.g., in CTS the sentence initial words do not vary as much as in BN).

6.3 Integration Methods for the LMs in an HMM

An important issue in combining multiple knowledge sources is the mechanism used for integration. Much depends on the knowledge sources and their individual models. For this work, we choose to train models using various knowledge sources separately, and then use a loosely coupled approach for model combination. As mentioned earlier, training a joint model of words and various class tags would require much more training data than is available for the structural event detection tasks. Given the annotated transcription, it is straightforward to train a word-based hidden event LM as described in Chapter 5. The word sequence W is then mapped to an automatically induced class sequence, a POS tag sequence P, and a chunk tag sequence CH (for BN only). Separate hidden event LMs are then trained from the various tag sequences plus the original event sequence E: AIC-LM, POS-LM, and chunk-LM. A word-based hidden event LM is also trained from the additional out-of-domain text corpora (word-LM-ood).

During testing, given a word sequence, the question is how to use various knowledge sources to make the best decision about the structural event. We how we com-

bine the various LMs first, and then integrate the combined LMs with the prosody model in the HMM system. Note that in the following descriptions, the LMs are all hidden event LMs.

- Combining with AIC-LM. For the automatically induced classes, since there is a one-to-one mapping from a word to its class, which can be done on the fly, The AIC-LM is combined with the word-based LM at the LM level. For example, for a word w_i and its history $H(w_i) = w_{i-n}^{i-1}$:

$$p(w_i|H(w_i)) = \alpha p_{word}(w_i|H(w_i)) + (1-\alpha)p_{AIC}(w_i|H(w_i)) \tag{6.7}$$

  where $0 < \alpha < 1$, and is optimized based on some held-out data set. p_{word} is the probability obtained from the word-based LM; p_{AIC} uses both the word and class information in a conditional modeling approach as shown in Equation (6.2):

$$P_{AIC}(w_i|H(w_i)) = p(w_i|C_i) \times P(C_i|C_{i-n}^{i-1}) \tag{6.8}$$

  This loosely coupled approach has fewer parameters than a joint model of classes and words.

- Combining with POS-LM and chunk-LM. Unlike the automatically induced classes, POS and chunk tags can not be obtained on the fly from the word sequence since it is not a one-to-one mapping and the entire word sequence is needed for tagging. First the word sequence is tagged to obtain the POS sequence using the TnT tagger, and then the chunks with the TBL text chunker. At each interword boundary, the most likely event E_i is computed according to the POS- or the chunk-based hidden event LMs individually. This is done using the same forward-backward algorithm as is used for the word-based LM. Then for each word boundary, the posterior probabilities from the POS- or chunk-based LM are combined via linear interpolation with the other LMs:

$$p(E_i|W) = \lambda p_{word}(E_i|W) + (1-\lambda)p_{class}(E_i|class). \tag{6.9}$$

where p_{class} can be p_{POS}, p_{chunk}, or p_{comb} depending on which LM is used, POS, chunk, or their combination. Note that unlike AIC-LM, word information is not used in POS-LM and chunk-LM.

- Combining with word-LM-ood. These LMs are trained from the out-of-domain (ood) corpora and are combined with the word LMs trained from the in-domain training data at the LM level. When the AIC-LM is also combined, the final combination becomes:

$$p(w_i|H(w_i)) = \alpha p_{word}(w_i|H) + \beta p_{AIC}(w_i|H) + (1-\alpha-\beta)P_{word-ood}(w_i|H) \tag{6.10}$$

where $0 < \alpha < 1$ and $0 < \beta < 1$, p_{AIC} is shown in Equation (6.8), and $p_{word-ood}$ means the word-based LM trained from the out-of-domain material. Combination at the LM level is implemented using functions in the SRILM toolkit [109].

Next we describe how these LMs are combined with the prosody model. Figure 6.1 shows the combination method. The box indicated by the dotted lines uses the HMM approach described in Chapter 4. Here rather than using the word-based hidden event LM, the state transition probabilities are obtained from the combination of the word-LM, AIC-LM, and the word-LM-ood (as in Equation (6.10)). These are then integrated with the prosody model as described in Chapter 4. The outputs from this box are the posterior probabilities for the events at each word boundary. The POS- and chunk-based LMs also generate posterior probabilities individually. Finally at each word boundary, all of the event probabilities from different models are interpolated, with the weights chosen using a held-out set.

6.4 Experiments on SU Detection Task

Various knowledge sources described above are evaluated, alone and in combinations for the SU boundary detection task in both CTS and BN. Training and test

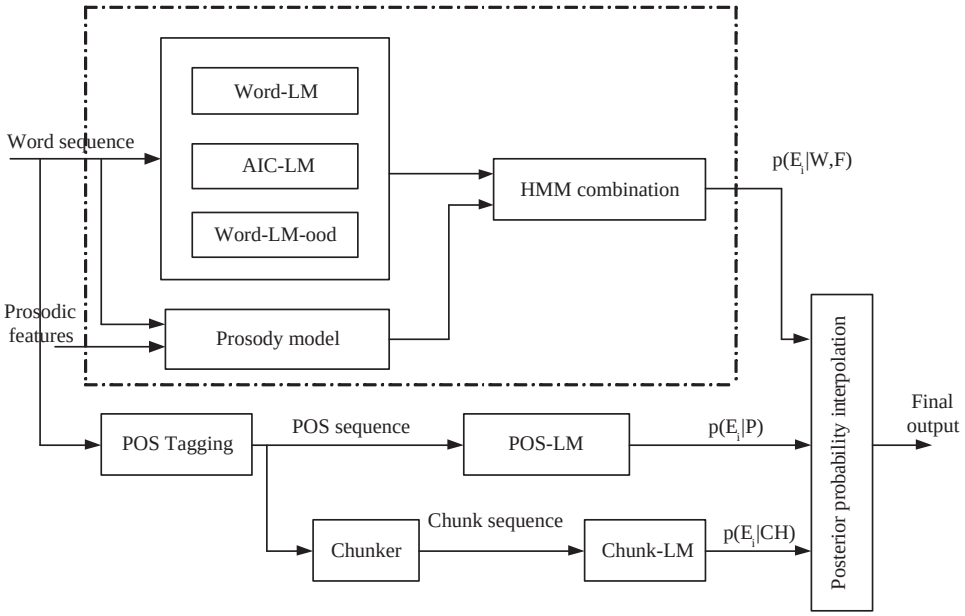

Fig. 6.1. Integration methods for the various LMs and the prosody model.

data are the same as for the baseline system. The annotated text corpus is used to train various class-based hidden event LMs: AIC-LM, POS-LM, and chunk-LM. The additional text material for CTS and BN is as described in Section 6.2.5.

6.4.1 CTS SU Task

Table 6.3 shows the results using various LMs for CTS SU boundary detection task. Results are only reported for human transcriptions so that we can more easily understand each model without considering the effect of incorrect words.

Among all of the LMs that are trained from the RT-03 training data, results show that when each is used alone, the word-based LM yields the lowest error rate, with the AIC-LM the second best, and the POS-LM coming last. The POS-LM loses much fine-grained lexical information when the word sequence W is converted to the POS sequence P. The AIC-LM performs similarly to the word-based LM. The vocabulary size of the AIC-LM (100 in our experiments) is not substantially greater

Table 6.3
SU detection results (NIST error rate in %) for human transcriptions of CTS data using various LMs, alone and in combination with the prosody model. The deletion (DEL), insertion (INS), and total error rate are reported.

CTS			
Model	**DEL**	**INS**	**Total Error Rate**
Word-LM	27.15	14.87	42.02
AIC-LM	30.86	14.45	45.31
POS-LM	37.17	23.57	60.74
Word-LM-ood	33.82	12.27	46.09
Word-LM+Prosody	18.86	17.38	36.24
Word-LM+Prosody + Word-LM-ood	19.06	15.99	35.05
Word-LM+Prosody +Word-LM-ood+AIC-LM	19.26	15.35	34.61
Word-LM+Prosody+Word-LM-ood +AIC-LM+POS-LM	19.39	14.71	34.10

than a POS-LM, yet it maintains some of the fine-grained information because it uses word information via $P(W|C)$ as in Equation (6.8). Another possible reason for this may be the poor POS tagging accuracy on CTS. Looking at the insertion and deletion errors, the AIC-LM and the word-LM have similar insertion errors. AIC-LM has slightly more deletions, possibly because some effective words that signal sentence start or end are not grouped correctly. POS-LM yields more insertion and deletion errors than other LMs.

The word-LM-ood performs worse than the in-domain word-LM and AIC-LM, but better than the POS-LM. There is some mismatch between that training corpus and the test condition; hence, even though the word-LM-ood is trained from

a relatively larger training corpus, it suffers from the mismatch. In addition, we find that the error pattern using the word-LM-ood is somewhat different from the word-LM trained from the RT-03 training data, i.e., there are fewer insertions and more deletion errors.

Results are also presented when the prosody model is combined with the various LMs. For SU detection, by combining various LMs with the baseline word-based LM and the prosody model, a consistent improvement is obtained compared to a word-based LM alone. We notice that compared to the results using the word-LM and the prosody model, combining with various additional LMs decreases the insertion errors, at the cost of the slightly increased deletion errors. This suggests that adding the prosody model tends to generate more insertion errors (see results for using word-LM versus word-LM+prosody); whereas, adding the various LMs tends to decrease the insertion errors and yield more deletion errors.

6.4.2 BN SU Task

Experimental results from using various LMs alone, and in combination with the prosody model, for the BN SU detection task are shown in Table 6.4. In contrast to what we observed on CTS, the POS-LM outperforms the AIC-LM on BN. One possible reason for this is that more training data is available to train the POS tagger, which is likely to result in a better tagging accuracy.[3] Another possible reason may lie in the difference between the language styles used on CTS and BN. Utterances tend to be more grammatical and more similar to written text on BN than for conversational speech. The chunk-LM alone does not perform well, probably because its coarse granularity does not preserve sufficient information to distinguish between events and non-events accurately.

Using the word-based LM trained from the out-of-domain corpus, the performance is similar to the word-based LM trained from the RT-03 data. The mismatch

[3]The tagging accuracy for these two corpora has been evaluated.

Table 6.4
SU detection results (NIST error rate in %) for human transcriptions of the BN data using various LMs, alone and in combination with the prosody model. The deletion (DEL), insertion (INS), and total error rate are reported.

BN			
Model	**DEL**	**INS**	**Total Error Rate**
Word-LM	66.00	14.44	80.44
AIC-LM	69.19	17.79	86.98
POS-LM	67.14	18.14	85.28
Chunk-LM	74.08	17.06	91.13
Word-LM-ood	66.23	13.87	80.10
Word-LM+Prosody	37.01	27.74	64.75
Word-LM+Prosody + Word-LM-ood	29.05	24.56	53.61
Word-LM+Prosody +Word-LM-ood+AIC-LM:	28.94	23.99	52.93
Word-LM+Prosody+Word-LM-ood +AIC-LM+POS-LM	29.90	22.51	52.42
Word-LM+Prosody+Word-LM-ood +POS-LM+Chunk-LM	31.84	20.41	52.25

between training and testing potentially affects the LM performance, but the availability of more training data helps address some of the sparse data problems and thus compensates in part for the mismatch between the training and testing sets. This is in contrast to what we have observed for CTS, where the word-LM-ood yields a lower accuracy than the word-LM trained from the RT-03 data. For CTS, the amount of

training data does not compensate for the mismatch between the LDC annotated training data and the additional text corpus.

It should be noted again that as on CTS, adding the prosody model decreases deletion error rate (although at the cost of some insertion errors). Combining with various LMs decreases SU detection error rate compared to the baseline system using the word-based LM and the prosody model. The contribution from the word-LM-ood on BN is more significant than that on CTS.

6.5 Chapter Summary

We have investigated using information beyond word-based knowledge, including automatically induced classes, POS tags, and chunk tags, together with the additional training data to improve the accuracy for SU detection. Experiments have confirmed that adding class-based LMs improves performance and that more training data is helpful. We observe the impact of the new sources differs across corpora. The POS-based LM performs relatively better on BN than on CTS, partly due to the better tagging accuracy, but largely because of the different speaking styles in the two corpora. The out-of-domain data is shown to be more important on BN than CTS possibly because it is relatively larger than the data size in CTS, and the data sparsity issue is more severe in BN than in CTS.

For the AIC-LM, a conditional modeling approach is used (as shown in Equation (6.8)) to combine word and class information, which can reduce model parameters compared to a word-based LM. For the POS- and chunk-based LMs, a loosely coupled model is used, i.e., first the class sequence for the test word string is generated, and then the event probabilities using the class token sequence are computed, which are then combined with the probabilities from the word-based LM. Heeman and Allen [56] proposed a tightly-coupled approach to find the best POS sequence and disfluency events together. Their experiments were conducted on the Trains corpus, which differs from the conversational speech in that it is far more template-based.

Given that there is interaction between syntactic information, SU boundaries, and disfluency phenomena, it is likely that a tightly-coupled model, such as the model proposed in [56], may perform better, but this would require a larger training set than is currently available. The maximum entropy approach that will be described in Chapter 8 provides an interesting alternative method for more tightly integrating knowledge sources.

7. PROSODY MODEL

As described in Chapter 4, for our baseline system, a decision tree classifier is used to implement the prosody model and then combine it with an N-gram language model in an HMM to find the most likely structural events at interword boundaries. Since the structural events are less frequent than the non-events in the training data, the decision tree prosody model must be designed to deal with the imbalanced class distribution. Our initial approach in the baseline system was to randomly downsample the training set to obtain a balanced data set for decision tree training in order to make the model more sensitive to the minority class (e.g., SU boundaries or IPs), followed by adjusting the posteriors in the test set. A problem with this approach is that many potentially important majority class samples are not used for model training, and thus downsampling may degrade performance on the test set. In this chapter, we investigate several sampling approaches to cope with the imbalanced class distribution, as well as a bagging scheme, in an attempt to build more effective structural event prosody model classifiers.

A pilot study is first conducted for the SU boundary detection task that uses a small training set in order to extensively evaluate all the methods. In this study, human transcriptions are used to factor out the effect of speech recognition errors. We also examine the effect of these approaches across different classification tasks (SU and IP detection) and evaluate the impact of training data size. Then based on the findings of this pilot study, some of the most successful methods are chosen to evaluate on the full NIST SU boundary task on both the CTS and BN corpora.

This chapter is organized as follows. Section 7.1 describes the imbalanced class problem and the approaches that are investigated to address this problem. In Section 7.2, different techniques are systematically evaluated for the SU boundary detection

task in a pilot study using the CTS corpus. In Section 7.3 the sampling and bagging approaches are investigated across the SU and IP tasks. Section 7.4 shows the results on the NIST SU task using some of the best approaches from the pilot study. Section 7.5 summarizes this chapter.

7.1 Addressing the Imbalanced Data Set Problem

7.1.1 The Imbalanced Class Distribution Problem

In a classification problem, the training set is *imbalanced* when one class is more heavily represented than the others. Clearly, this problem arises in our structural event detection tasks, as we have mentioned earlier: only about 13.6% of the inter-word boundaries correspond to SU boundaries in conversational speech, and 8.1% in broadcast news speech. The data is even more skewed for the IP detection with only about 4.5% of the boundaries being IPs in CTS, and 1.1% in BN.

The imbalanced data set problem has received much attention from statisticians and the machine learning community [110–119]. Various approaches attempt to balance the class distribution in the training set by either oversampling the minority class or downsampling the majority class. Some variations on these approaches use sophisticated ways to choose representative majority class samples (instead of randomly choosing some of the majority class samples to match the size of the minority class), to synthetically generate additional samples for the minority class (rather than replicating the existing samples), or to combine classifiers trained from both downsampled and oversampled data sets. It is important to note that most of these techniques focus on improving the minority class prediction (due to its relatively higher importance in their problem specification). For example, in application like fraud detection or tumor detection, the minority class is clearly more important. Weiss and Provost [120] observed from their empirical study that the naturally occurring distribution is not always the optimal distribution and that when using receiver operating characteristics (ROC) as a performance criterion, a balanced distribution

is often a preferred choice. While sampling methodologies generally improve the prediction of the minority class, they also tend to penalize the majority-class cases. However, for the structural event detection tasks investigated in this thesis, both false positives and false negatives are considered equally costly.[1] Therefore, changing the distribution to have relatively more minority class samples may not produce a classifier with the best performance. Our goal is thus to evaluate various techniques to address the imbalanced class distribution in the training sets for structural event detection.

Which sampling method is the best greatly depends on the properties of the application, such as how the samples are distributed in the multidimensional space and the extent to which the different classes are mixed. Therefore, a systematic investigation of different sampling approaches is important for building better models.

In addition to sampling methods, we investigate the use of bagging. Bagging samples the same training set multiple times and has been shown to outperform a single classifier trained from the training set [121]. Both sampling and bagging techniques are described in the next section.

To our knowledge, this is the first study on the imbalanced class problem for structural event detection based on speech inputs. This study should provide groundwork for future classification tasks related to spoken language processing, such as finding hot spots in the meeting corpus [122], where classes are also imbalanced. The present study has properties that are characteristic of machine learning tasks for speech: it involves rather large amounts of data, it involves inherent ambiguity (SU boundaries and IPs are sometimes a matter of judgment), the data is noisy because of both measurement errors (from imperfect forced alignments and pitch extraction) and labeling errors (human labelers make errors), and the class distribution is heavily skewed, the latter being the main issue addressed in this chapter. In addition, the property that the majority and the minority classes are of equal interest is another attribute that

[1]More studies are needed to see whether missing a structural event and inserting an incorrect event have the same impact on human understanding or downstream language processing modules. Currently, equal penalty is used for both insertion and deletion errors in the scoring procedure.

makes this problem interesting. We believe that this study is therefore beneficial to both the machine learning and the speech and language processing communities.

7.1.2 Approaches to Address the Problem

Sampling Approaches

In a pilot study, four different sampling approaches are investigated, as well as no sampling (i.e., the original training set is used):

- Random downsampling: This approach randomly downsamples the majority class to equate the number of the minority and majority class samples. Since this method uses only a subset of the majority class samples, it may result in poorer performance for the majority class [111, 114, 115].

- Oversampling using replication: This sampling approach replicates the minority class samples to equate the number of the majority and minority class samples. All the majority class samples are preserved, and the minority class samples are replicated multiple times. The replication of the poor minority class samples for addition to the training set may lead to poorer performance of the minority class [111, 115, 119].

- Ensemble downsampling: Ensemble downsampling is a simple modification of random downsampling, in which the majority class is split into N subsets, each with roughly the same number of samples as the minority class [118], and then the classifier is trained using these subsets together with the minority class. In the end, there are N decision trees, each of which is trained from a balanced training set. For testing, the posterior probabilities from these N decision trees are averaged to obtain the final decision. The samples used for this approach are the same as in the oversampling approach, that is, all the majority class samples are used plus the minority class samples that are replicated N times. The two approaches differ only in how the decision trees

are trained. The ensemble downsampling approach is more scalable since it can be easily implemented in a distributed fashion.

- Oversampling using synthetic samples — SMOTE:SMOTE stands for 'Synthetic Minority Over-sampling Techniques'. In the oversampling approach, the minority class samples are replicated multiple times. By contrast, the SMOTE [111] approach generates synthetic minority class samples rather than replicating existing samples. Synthetic samples are generated in the neighborhood of the existing minority class examples. For continuous feature values, SMOTE produces new values by multiplying a random number between 0 and 1 with the difference between the corresponding feature values of a minority class example and one of its nearest neighbors in the minority class. For nominal cases, SMOTE takes a majority vote among a minority class example and its k-nearest neighbors. The synthetic samples can potentially cause the classifier to create larger and less specific decision regions, which can generalize better on the testing set than simple oversampling with replication.

- Original data set: There is no sampling in this method. The original training set is utilized as is.

The Bagging Technique

Bagging [121] combines classifiers trained from instances that are sampled with replacement given a training set. The bagging algorithm is shown in Figure 7.1. To maintain a fixed class distribution for all the bagging trees, each class is sampled separately. Therefore, in each bag the class distribution is the same as in the original given data set. T sets of samples are generated, each of which is used to train a classifier, and the final classifier is built from the T classifiers, equally weighted. Since each classifier generates a posterior probability for a test sample, the outputs from these classifiers can be averaged to obtain the final probability for this sample.

Input: training set S, number of bagging T

Bagging (T, S)
size_1 = sizeof (class 1 in S)
size_2 = sizeof (class 2 in S)
for $i = 1$ to T {
    S'_1 = sample from class 1 in S (with replacement) size_1 times
    S'_2 = sample from class 2 in S (with replacement) size_2 times
    $S' = S'_1 + S'_2$
    train a decision tree C_i from S'
}

Output: T classifiers

Fig. 7.1. The bagging algorithm. T is 50 in our experiments. In each bag, the class distribution is the same as in the original data S.

Bagging has several advantages. First, because different classifiers make different errors, combining multiple classifiers generally leads to a superior performance when compared to using a single classifier [123]. The combination of multiple trees from different bags of sampled instances makes the final classifier more noise tolerant. Second, bagging can be implemented in a parallel or distributed fashion to speed up training time [124]. Finally, bagging is able to maintain the class distribution of the training set on which bagging is applied. This is important since when the prosody model is combined with the LM as described in Chapter 4, it is easier to have a fixed class distribution in the training set. One disadvantage of bagging is that bagging results in multiple decision trees that can make it more difficult to understand the features that contribute most to a final decision.

7.2 Pilot Study for SU Detection

7.2.1 Experimental Setup

Features and Data Set

In this pilot study, a small subset of the RT-03 CTS training data is used in order to evaluate each of the methods described above. Table 7.1 describes the data set that is used in this pilot study. The training data set contains about 128K word boundaries, of which 16.8K are in the SU class, with the remaining being non-SUs. The test set consists of 16K word boundaries.

Because it is time consuming to train a decision tree with a large number of features and also to synthetically generate minority samples using the SMOTE approach, first we trained a decision tree from a downsampled training set using all the prosodic features in Section 4.2, and then used the features selected by this decision tree (25 features in total) to evaluate the various sampling approaches in order to minimize the computational effort for the pilot work. For these initial investigations, human transcriptions are used instead of the speech recognition output to factor out the impact of recognition errors on our investigation of the prosody model.

Table 7.1
Description of the data set used in the pilot study for the CTS SU detection task.

Training size	128K
Test size	16K
Class distribution	87% are SUs, 13% are non-SUs
Features	25 features (2 discrete)

Evaluation Conditions

All of the sampling and bagging approaches are evaluated on the test set under two conditions:

- Prosody model alone. For the decision trees trained from a balanced data set, the priors and the posterior probabilities generated by the decision trees are combined to obtain the final hypothesis for the imbalanced test set as shown in Equation (5.1). For the decision trees trained from the original data set, the posterior probabilities do not need to be adjusted, under the assumption that the original training set and the real test set have similar class distributions.

- Combination with LM. The combination of the prosody model and the hidden event LM is evaluated on the test set. If the decision tree is trained from the balanced data set (no matter which downsampling or oversampling approach is used), the posterior probability from the decision tree needs to be adjusted, finally resulting in Equation (5.3) for the HMM. For the decision tree trained from the original data set, the posterior probability generated by the decision tree is the true posterior probability $P(E_i|F_i, W_t)$ in the numerator of Equation (4.8). Therefore, the priors of different classes must be taken into account as shown in Equation (4.8).

Results are reported using the classification error rate (CER), F-measure, and the ROC and AUC metrics described in Chapter 3. These metrics are used in order to better focus on the machine learning aspects of the problem. For all of the test conditions, a β of 1 is used in the F-measure computation. In addition, a threshold of 0.5 is set to make the final decision at each boundary, that is, when the posterior probability of being an SU boundary is greater than 0.5, then it is hypothesized as an SU; otherwise, it is a non-SU boundary.

7.2.2 Sampling Results

Experimental results for the different sampling approaches are shown in Table 7.2. Generally, the downsampling methods outperform the oversampling method that employs replication, which can lead to overfitting. As can be observed from the table, the CER from oversampling by replication is much higher than any other techniques in the table, thus indicating that the decision tree does not generalize well on the testing set. Downsampling is a good way to increase the sensitivity of the decision tree classifier to the minority class. There is a slight improvement when using an ensemble of multiple decision trees over using a single randomly downsampled data set to train the prosody model. However, this gain does not hold up when combining the prosody model with the LM. This suggests that even though one classifier alone achieves good performance, other knowledge sources (e.g., a language model) may mask this gain.

Table 7.2
SU detection results (CER in % and F-measure) for different sampling approaches in the pilot study of the CTS corpus, using the prosody model alone and in combination with the LM. The CER of the LM alone on the test set is 5.02%.

Approaches	Prosody alone		Prosody + LM	
	CER	F-measure	CER	F-measure
Chance	13	0	-	-
Downsampling	8.48	0.612	4.20	0.837
Oversampling	10.67	0.607	4.49	0.826
Ensemble	7.61	0.644	4.18	0.837
SMOTE	8.05	0.635	4.39	0.821
Original	7.32	0.654	4.08	0.836

Using the original training set achieves the best overall performance in terms of CER. This is potentially due to the equal costs assigned to both the classes. It is also possible that there are sufficient examples belonging to the minority class in the training set to learn about the SU boundaries. Thus, the CER is lower than the other sampling approaches. However, training a decision tree from the original training set takes much longer than from a downsampled training set. Also if the training set were large or heavily skewed, the advantage of the original training set may be diminished.

SMOTE improves upon the results from both the downsampling and the oversampling approaches for the prosody model alone. SMOTE introduces new examples in the neighborhood of the minority class cases, thereby improving the coverage on the minority class cases. Since SMOTE enables the entire majority class set to be used in a single decision tree, it can also improve the performance on the majority class (i.e., the non-SU decision). However, SMOTE can lead to a computational bottleneck for very large data sets, in which the distribution is imbalanced but there are sufficient examples belonging to the minority class. More examples are added to the original training set, and if it is already very large, it can substantially increase training time. One can possibly deploy a combination of SMOTE and downsampling to counter the large training set size [111]. Notice also that the gain from the SMOTE method when the prosody model is used alone does not hold up when it is combined with the LM. This may be due to the fact that the synthesized samples to some extent are incompatible with what normally happens in language, or the samples that SMOTE helps make correct decisions for the prosody model alone are the ones that are already well modeled by the LM.

Figure 7.2 compares the various techniques using the ROC curves and the AUC obtained by each of the approaches using the prosody model alone. The ROC curves span over the entire continuous region of classification thresholds, and hence provide a visualization of the trade-off between the true positives and false positives. For the CER measurement, using the original training set achieves the best performance

(as shown in Table 7.2); however, the advantage of the sampling techniques is more pronounced when we look at the ROC curves (and the corresponding AUC value). The AUC of the sampling and ensemble techniques is significantly larger than the AUC obtained by training a decision tree on the original distribution. Like Weiss and Provost [120], we observe that downsampling beats oversampling with replication, while SMOTE beats both oversampling and downsampling. This has also been observed by other researchers in the machine learning literature [125, 126]. As shown in Figure 7.2, at the lower false positive (FP) rate, the original distribution is competitive to the sampling techniques, while at higher FP rates, the sampling schemes significantly dominate over the original distribution. If the minority class were of greater importance, then one would tolerate more false positives and achieve a higher recognition on the minority class by locating the appropriate operating point. If obtaining a high recall for the SU detection task is important, then based on the ROC analysis, the sampling techniques are definitely useful. The non-smooth ROC using the original training set is largely due to its imperfect probability estimation. For example, the minimum posterior probability among all the test samples is 0.16 according to the decision tree; therefore, when the decision threshold is greater than 0.16, then all the test samples are hypothesized as being in the positive class, resulting in a sharp turning point. Whereas, in the other sampling approaches, the posterior probabilities span over the entire region between 0 and 1.

In Table 7.3, to focus on the error patterns for each sampling method, we show the precision and recall rate using the prosody model alone, as well as in combination with the LM. Using the prosody model alone, oversampling yields the best recall result, at the cost of a lower precision. This may result from replicating the minority class samples multiple times. We had expected that using a balanced training set would be beneficial to the recall rate; however, contrary to our expectations, the recall performance from the downsampling and ensemble sampling approaches is not better than using the original training set. Note again that 0.5 is used as the threshold to make decisions based on the posterior probabilities, since the false positive and the

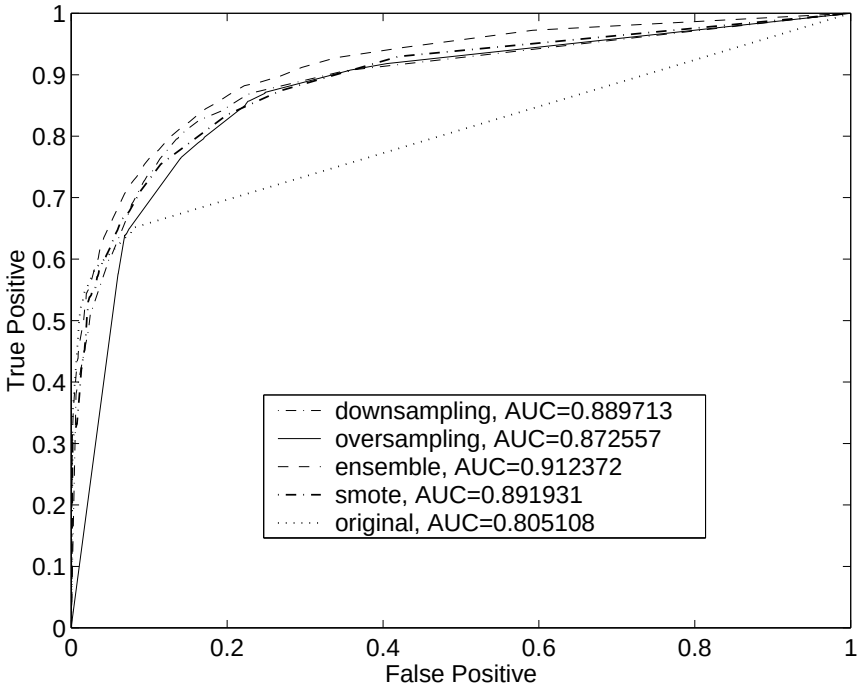

Fig. 7.2. ROC curves and their AUCs for the decision trees trained from different sampling approaches and the original training set.

false negative errors are equally costly. Thus, there are many fewer false-positives if the original distribution is used in learning the decision tree. This leads to a much higher value of *precision* compared to any of the sampling techniques. After the prosody model is combined with the LM, we observe that the recall rate is substantially improved for the downsampling and ensemble sampling approaches, resulting in a better recall rate than when the original training set is used. However, SMOTE does not combine well with LM: the recall rate is the worst after combining with LM even though SMOTE yields a better recall rate than downsampling or ensemble sampling when the prosody model is used alone. The gain in the recall rate from the oversampling approach when the prosody model is used alone is diminished, too, when combined with LM.

Table 7.3
Recall and precision results for the sampling methods in the pilot study of CTS SU detection. Using LM alone yields a recall of 74.6% and a precision of 84.9%.

Approaches	Prosody alone		Prosody + LM	
	Recall	Precision	Recall	Precision
Downsampling	51.4	75.6	83.0	84.5
Oversampling	63.5	58.2	82.2	83.1
Ensemble	53.2	81.4	82.6	84.9
SMOTE	53.8	77.4	77.4	87.4
Original	53.5	84.7	80.0	87.5

7.2.3 Bagging Results

Several sampling techniques have been selected on which bagging is applied. Since downsampling is an approach that is computationally efficient and does not significantly reduce classification accuracy, first bagging is applied to a downsampled training set to construct multiple classifiers. Bagging together with the ensemble approach is also tested. As described above, for the ensemble approach, the majority class samples are partitioned into N sets, each of which is combined with the minority class samples to obtain a balanced training set for decision tree training. The final classifier is the combination of the N base classifiers. Bagging (with trial number T) is applied to each of N balanced sets, and thus $T \times N$ classifiers are generated for combination. Finally, bagging is applied to the original training set. 50 bags are used for each of the bagging experiments. We do not combine bagging with any oversampling approaches because of their poorer performance compared to downsampling or using the original training set.

The bagging results are reported in Table 7.4. The table shows that bagging always reduces the classification error rate over the corresponding method without

Table 7.4
CTS SU detection results (CER in % and F-measure) with bagging applied to a randomly downsampled data set (DS), ensemble of downsampled training sets, and the original training set. The results for the training conditions without bagging are also shown for comparison.

Approaches	Prosody alone		Prosody + LM	
	CER	F-measure	CER	F-measure
Downsampling	8.48	0.612	4.20	0.837
Ensemble downsampling	7.61	0.644	4.18	0.837
Original	7.32	0.654	4.08	0.836
Bagging on DS	7.10	0.665	3.98	0.845
Bagging on ensemble DS	6.93	0.673	3.93	0.847
Bagging on original	6.82	0.676	3.87	0.849

bagging. Bagging the downsampled training set uses only a subset of the training samples, yet achieves better performance than using the original training set without bagging. The difference between bagging on the original training set and ensemble bagging is not significant (i.e., $p > 0.05$ using the sign test). Bagging is able to construct an ensemble of diverse classifiers, and improves the generalization of decision tree classifiers; it mitigates the overfitting of a single decision tree classifier. The gain is more substantial when bagging is applied to the downsampled training set than to the original training set or the ensemble sampling sets, compared to each of the corresponding conditions without bagging respectively.

Similarly to the study on sampling techniques, the ROC curves are plotted for the three bagging schemes in Figure 7.3, with a zoomed version of the curves shown at the bottom. The AUC is substantially better when bagging is employed compared to the results shown in Figure 7.2, and the three bagging curves are very similar. Notice that the AUC is improved substantially when bagging is applied to the original training set. This is attributed to the better posterior probability estimation, which

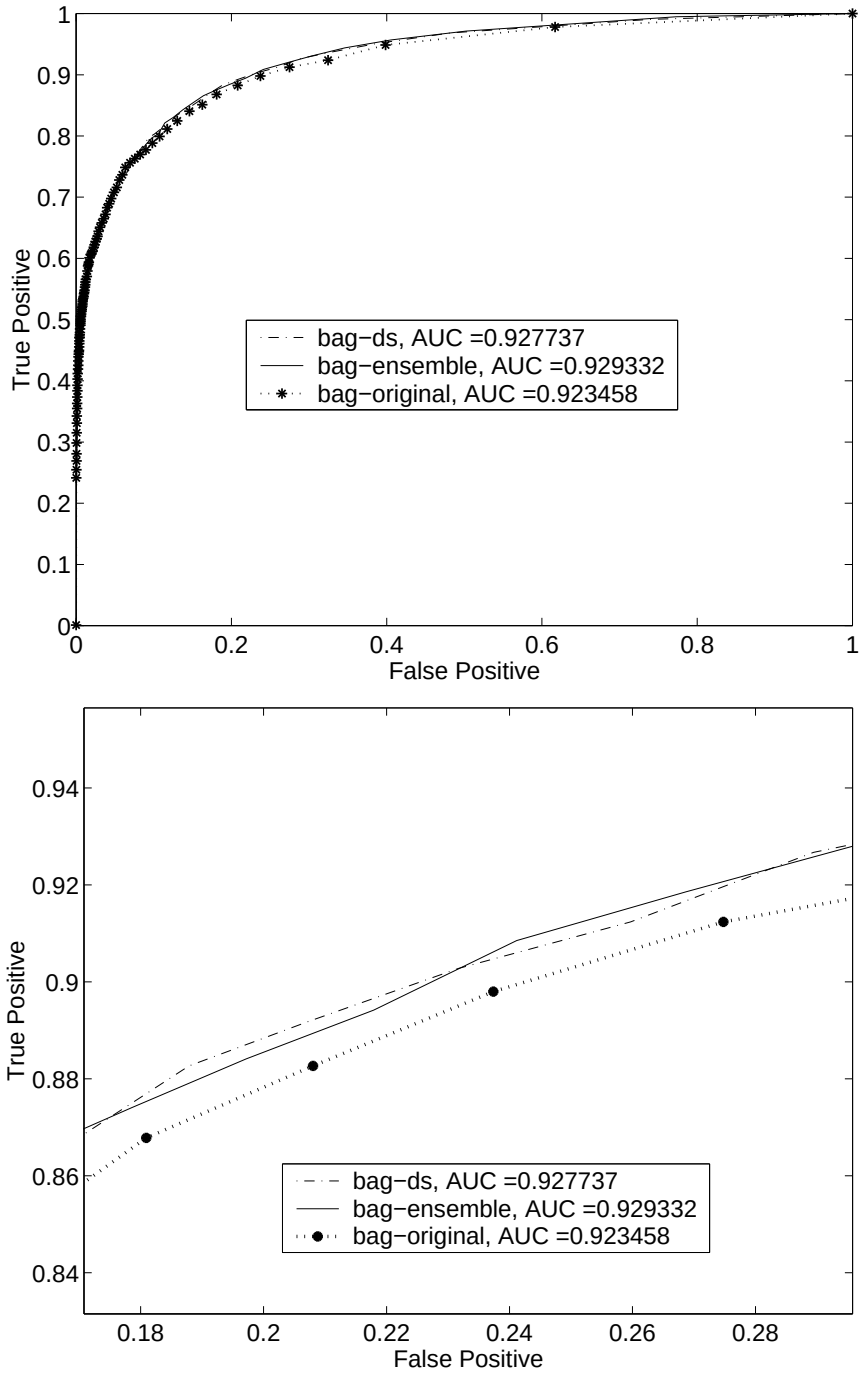

Fig. 7.3. ROC curves and their AUCs for the decision trees when bagging is used on the downsampled training set (bag-ds), the ensemble of downsampled training sets (bag-ensemble), and the original training set (bag-original).

is obtained from the average of multiple classifiers. This explains why the ROC curve is more smooth with bagging applied to the original training set compared to the curve without bagging shown in Figure 7.2. Consistent with the results without bagging, applying bagging to the original training set yields a slightly poorer AUC than for the downsampled and ensemble bagging cases.

7.3 Sampling and Bagging Across SU and IP Tasks

We have investigated a variety of sampling approaches, as well as bagging and ensemble methods for detecting SU boundaries. Considering that these approaches may be affected by the characteristics of this task, for example, the data is less skewed than for other structural events and the prosodic features are quite informative, we also evaluate the techniques on a different task, IP detection. Additionally, the effect of the data size is examined for the SU detection task.

7.3.1 Experimental Setup

Data

In this experiment, the full RT-03 CTS training set is used, rather than the pilot data due to the relative infrequency of the IP event (see Table 3.2). The 754 conversations are split into training and testing sets. Table 7.5 shows the experimental setup, including the training and testing set sizes (number of inter-word boundaries) and the percentage of the minority class in the data set for the task of IP and SU detection. Since this data set is larger than the one used for the pilot SU study, this also allows us to examine the effect of the data size on the sampling and bagging techniques for the SU task. For comparison, we include the description of the data set used in the pilot study for the SU task in Table 7.5.

Table 7.5
Description of the data sets used for the SU and IP detection tasks. The data set used in the pilot study is shown in the second column, which is a subset of the data set used in this investigation ("large set" denoted in the table).

	SU		IP
	Pilot data	Large set	Large set
Training set	128K	428K	428K
Test set	16K	53K	53K
Percentage of the minority event	13.0	13.56	4.54

Methods

Similarly to the pilot study, a decision tree was first used to choose some important features, which are then used for the other sampling and bagging techniques. This minimizes computational effort. For both the SU and IP tasks, evaluation is conducted on the reference transcription condition, and performance is measured using CER.

Among the sampling and bagging approaches evaluated in the pilot study, SMOTE is not used due to its computational complexity and lack of performance gain. Hence, that leaves the following approaches for the study across SU and IP tasks: original training set, downsampling, oversampling, ensemble sampling, bagging on downsampled set, and bagging on ensemble sampling.

7.3.2 Results Across SU and IP Tasks

Table 7.6 shows the experimental results of the sampling and bagging approaches for the IP and SU detection tasks. In addition to evaluating on the original test set, we present the results on a downsampled test set when using the prosody model

alone for IP detection, due to the fact that the prosody model does not do better than chance performance on the non-downsampled test set.

Table 7.6
IP and SU detection results in CER (%). 'DS' denotes 'downsampled'. Chance performance is 4.36% on the original test set for IP, and 13.64% for SU. The CER using LM alone is 2.34% on the IP task, and 5.27% on the SU task.

Method		IP			SU	
		DS test set	original test set		original test set	
		Prosody	Prosody	Prosody+LM	Prosody	Prosody+LM
sampling	original	50	4.36	2.34	7.45	4.53
	downsampled	23.76	4.36	2.27	8.05	4.42
	oversampled	27.69	4.36	2.31	8.46	4.64
	ensemble	22.07	4.36	2.24	7.86	4.47
bagging	on DS	20.64	4.36	2.25	7.26	4.29
	on ensemble	20.20	4.36	2.24	7.22	4.35

Effect of Data Size for the SU Task

First the impact of the data size for the SU task is examined by comparing the results in Table 7.6 and Table 7.2. As the data set size increases, we expected that the gain from using the original training set might be lost and the benefit from ensemble sampling might decrease, since a downsampled training set might be more representative of the data set. Table 7.6 shows that contrary to our expectation, using the original training set yields the best results, although it has a greater cost in training time. As expected, the gain from ensemble sampling is diminished as the data set size increases. When the data set is small, ensemble sampling has the advantage of making use of the full data set within the ensemble. As the data set increases and is inherently more representative, the benefit from ensemble decreases. Similar to using the smaller data set in the pilot study, oversampling is computation-

ally expensive and does not yield a performance improvement. Downsampling the training set performs reasonably well, and has the advantage of saving computation. This is important when the training set size is large, i.e., hundreds of thousands of data samples. For both the data sets, which differ in data size, bagging outperforms a single classifier trained without bagging.

Sampling and Bagging Results for the IP Task

We observe from Table 7.6, using the prosody model alone on the original test set, none of the approaches (original training set, downsampling, bagging, ensemble) is able to win over the bias of the majority class and achieve performance better than the baseline chance performance.

If the original training set is used, then because the IP samples are an extremely small portion of the training set, the decision tree does not split, i.e., the classifier is not able to learn the characteristics of the minority class. Therefore, the classifier performs at chance on a downsampled test set. Sampling methods, which use a more balanced training set, improve classification performance on a downsampled test set. Bagging and ensemble bagging perform significantly better than the corresponding approaches without bagging on the downsampled test set when using the prosody model alone.

The prosody model trained from the original data set does not provide any information when combined with the LM on the original test set; whereas, for the other techniques (sampling and bagging), despite achieving only chance performance when used alone, the prosody model provides added information after it is combined with the LM.

Comparisons Between the IP and SU Tasks

Since IPs are much less frequent than SU boundaries (4.5% vs. 13.5%), sampling appears to have a different impact on the two tasks. On the SU detection task,

for the prosody model alone, the best performance is achieved by using the original training set among the different sampling approaches; whereas, for the IP task, none of the sampling approaches is able to yield better performance than chance. On a downsampled test set of the IP task, when the prosody model is used alone, sampling techniques help improve classification performance. When the prosody model is combined with the LM, the relative error rate reduction compared to using LM alone is smaller for the IP detection task than for the SU detection task, i.e., 4.3% versus 18.6% respectively, when using the prosody model trained from bagging on downsampled data set. When the prosody model is used alone, bagging improves classification accuracy on a downsampled test set for the IP task; however, on the non-downsampled IP test set, only chance performance is achieved. In contrast, for the SU task, bagging yields a substantial gain even on the non-downsampled test. When combined with the LM, there is a gain from bagging on both the SU and IP tasks.

Figure 7.4 shows the ROC curves for the IP and SU detection tasks for the original test set using the downsampled training set, bagging on downsampled training set, and ensemble bagging when using the prosody model alone. These curves suggest that bagging indeed improves the performance over using a single randomly downsampled training set. The ROC curve from ensemble-bagging is similar to that using bagging on one downsampled set. Notice also that the relative improvement from bagging on the IP detection task is larger than on the SU task (by looking at the improvement from the two ROC curves), suggesting that bagging improves the generality of decision tree classifiers more on the IP task than on the SU task.

7.4 Evaluation on the Full NIST SU Task

7.4.1 Experimental Setup

Some of the best techniques identified by the pilot study are evaluated on the full NIST SU detection task using the CTS and BN corpora, on both the reference and

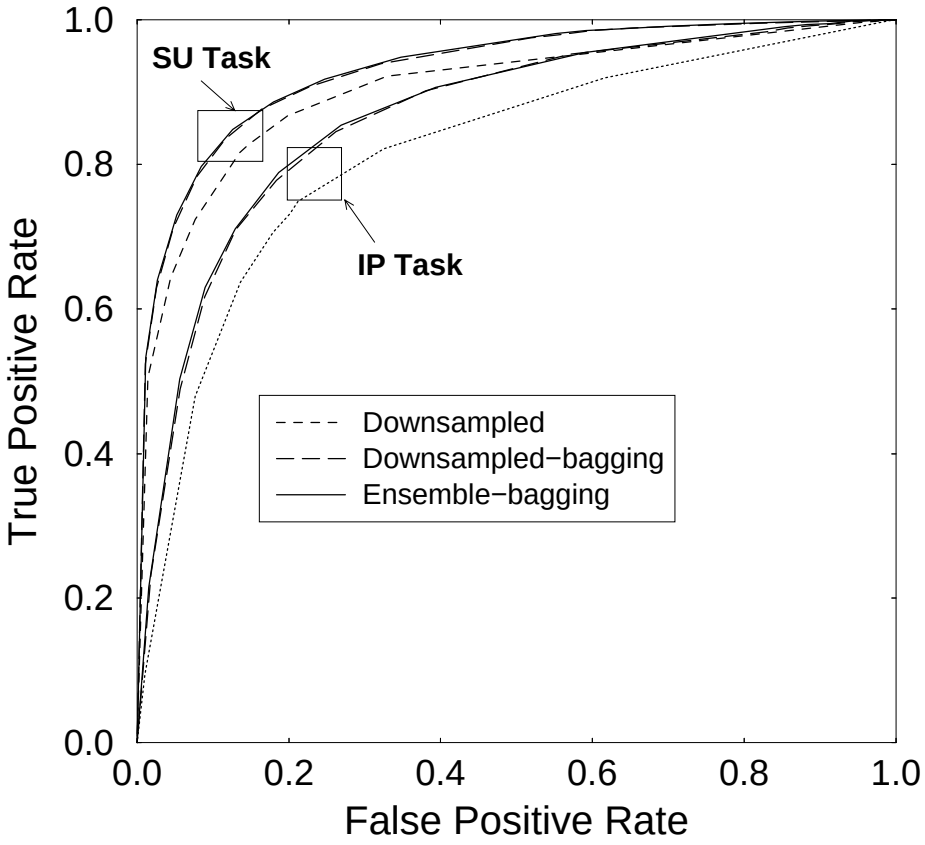

Fig. 7.4. ROC curves for IP and SU detection using the prosody model alone on the CTS corpus.

the STT output conditions. Data is the full set that is used in the previous chapters (see Table 3.2). Results are reported using the official NIST SU error rate metric, in order to compare with our baseline systems in Chapter 5. In addition, evaluation is conducted not only on the reference transcriptions, but also on the STT output, which makes using the classification error rate less straightforward.

Results of the pilot study suggest that bagging is beneficial for generating a robust classifier, and the two best approaches are ensemble bagging and bagging on original training set. Hence, we will evaluate these two approaches, alone and in combination with LMs. Since a downsampled training set was trained in the baseline system in Chapter 5, this is also included for comparison. In contrast to the pilot study, we

preserve all the prosodic features (total 101 features), expecting that bagging could generate different trees, each of which might use different features.

7.4.2 Results on the NIST SU Task

Table 7.7 shows the results using each of the above prosody models, and their combination with LMs on the CTS and BN SU tasks, using both the reference transcription (REF) and the recognition output (STT). The LMs used are the word-based hidden event LMs, trained from the LDC annotated training data plus the out-of-domain extra text corpora as described in Chapter 6. Overall, there is a performance degradation when using the speech recognition output, as we observed in Chapter 5. Recognition errors affect both the LMs and the prosody model, with less impact on the latter. The gain from bagging and sampling techniques in the reference transcription condition seems to transfer well to the STT condition. In both conditions, we find that applying the bagging technique yields a substantial win compared to the non-bagging conditions. When the prosody model is used alone, applying bagging on the original training set achieves significantly better results (at $p < 0.05$) than ensemble bagging on both of the corpora; whereas, when the prosody model is combined with LMs, the difference between using bagging on the original training set and bagging on the ensemble of balanced training set is diminished (i.e., the gain is not significant).

Even though the pilot study is only conducted using the CTS corpus, results in Table 7.7 shows a similar trend for the BN corpus. Although there are some differences across the two corpora (e.g., different class distributions because of their different speaking styles), the gain from bagging on the original training set is also observed on BN and is in fact greater than that found in CTS. Since the prosody model contributes relatively more for the BN corpus than CTS, a better prosody model is relatively more beneficial to BN.

Table 7.7
SU detection results (NIST error rate in %) for both the CTS and BN corpora, on the REF and STT conditions.

Approaches	BN		CTS	
	REF	STT	REF	STT
LMs	68.16	72.54	40.56	51.85
Prosody-downsampling	85.67	85.67	68.77	70.98
Prosody-ensemble-bagging	72.94	72.09	61.23	64.35
Prosody-bagging-original	67.65	67.77	59.19	62.98
LMs + prosody-downsampling	53.61	59.69	35.05	45.30
LMs + prosody-ensemble-bagging	50.03	56.17	32.71	43.71
LMs + bagging-original	49.57	55.14	32.40	43.81

7.5 Chapter Summary

7.5.1 Summary

We have attempted to build more robust prosody model in this chapter by addressing the imbalanced data set problem that arises in the structural event detection tasks. Several sampling and bagging approaches are investigated for training the decision tree prosody model. Empirical evaluations in a pilot study for the SU detection task show that downsampling the data set generates a reasonably good classifier, while requiring less training time. This computational advantage might be more important when processing a very large training set. Oversampling with replication increases training time without any gain in classification performance. An ensemble of multiple classifiers trained from different downsampled sets yields performance improvements when using the prosody model alone for the SU task. We have also found that the performance of the prosody model alone may not always be correlated with results obtained when the prosody model is combined with

the language model; for example, SMOTE outperforms the downsampling approach when the prosody model is used alone, but not when the prosody model is combined with the language model. Using the original training set achieves the best classification error rate among the sampling methods. However, if ROC or AUC is used, then using a balanced training set yields better results than the original training set, especially if the minority class is of more interest.

Bagging was investigated on a randomly downsampled training set, an ensemble of multiple downsampled training sets, and the original training set. Bagging combines multiple classifiers and thus it reduces the variance caused by a single classifier and improves the generality of the classifiers. Bagging results in even better performance than the use of more samples (e.g., comparing bagging on a downsampled training set versus the original training set without bagging) when using the prosody model alone for the pilot study of the SU task. Bagging can run in parallel, and thus training is computationally efficient.

Our investigation of the IP task highlights differences between the IP and SU tasks, which are probably due to differences in the magnitude of skew or inherent differences in the cues to these two different phenomena. When the prosody model is combined with the LM, we found that sampling techniques are more important in the case of the IP task, which has a much more severe problem with data skew. Bagging generates more robust classifiers for both SU and IP detection. A preliminary experiment using boosting also highlights additional differences between the SU and IP detection tasks (see Appendix A).

Several of the best methods found from the pilot study have been evaluated in the NIST SU detection task across CTS and BN corpora and across transcriptions. Bagging has yielded substantial gain compared to the baseline system. Additionally we find that when the prosody model is used alone, significantly better performance is observed when bagging is used on the original training set than ensemble bagging, yet most of the gain is eliminated when the prosody model is combined with the LM for SU detection.

7.5.2 Discussion

These experimental results confirm that using multiple classifiers reduces the variance and improves the robustness of the model. The combination of multiple learned models has been a research topic in the machine learning community, with the goal of forming an improved estimate [127, 128]. There are two issues involved in model combination.

The first issue is model generation. It is important to generate a set of models that are diverse in the sense that they make errors in different ways. Different approaches have been developed for generating multiple models. One approach is to use a particular learning algorithm and a data resampling technique to create a set of learned models. Bagging is such a technique to obtain multiple classifiers by using different resampling of the training data. Another approach is to use a variety of learning algorithms on all of the training data. Additionally, multiple models could also be learned using different feature sets. These techniques attempt to achieve diversity in the errors of the learned models by varying training data, learning algorithms, or features. The multiple models are typically combined using variants of a weighted majority strategy. Combining multiple classifiers is an interesting future direction for our investigation of more effective prosody model. Our experiments so far have not yielded any performance gain from the combination of multiple decision trees that are trained using different features sets, or the combinations of various sampling approaches as used in this chapter.

The second issue is model combination, i.e., to decide which models to rely on for a decision and how much weight to give each model. When the errors made by different models are uncorrelated, taking the majority vote is a reasonably good approach for their combination. However, when patterns exist in the errors that different models make, a more elaborate strategy could be far better. For example, a combining method needs to identify the unique contribution of each model and the inherent redundancy among them. Currently we have only used a simple average

of the posterior probabilities from a variety of decision trees for the prosody model. Training a super classifier that learns how to combine multiple classifiers is clearly an important goal for future work in this area.

8. APPROACHES TO COMBINE KNOWLEDGE SOURCES

As described in previous chapters, our baseline structural event detection system is based on an HMM approach. While an HMM is computationally efficient and provides a convenient way for modularizing the knowledge sources, it has two main drawbacks. First, the standard training methods for HMMs maximize the joint probability of the observations and the hidden events, as opposed to the posterior probability of the correct hidden variable assignment given the observations. The latter is a criterion more closely related to classification error. Second, the N-gram LM underlying the HMM transition model makes it difficult to use features that are highly correlated (such as word and POS labels) without greatly increasing the number of model parameters; this in turn makes robust estimation difficult. In Chapter 6, a non-optimal approach (i.e., interpolation at the LM level or posterior probability level) has been used for combining different textual sources within the HMM framework.

In this chapter, we describe our efforts to overcome some of the shortcomings of an HMM by using a maximum entropy (Maxent) classifier and the conditional random field (CRF) sequence decoding method for the SU detection task. These approaches estimate the conditional posterior probabilities directly, in contrast to the generative HMM. They also provide a more principled way to combine a large number of overlapping features. The Maxent and CRF approaches differ in that the former does not directly model sequence information. Although both techniques have been used previously for some traditional NLP tasks, they have not been widely applied to a task with both prosodic and textual information, as used in the SU detection task. We describe the techniques that have been developed to incorporate these

knowledge sources, and compare SU detection performance using the HMM, Maxent, and the CRF approaches on two different genres of speech: CTS and BN. How word recognition error and different knowledge sources affect the comparison will also be investigated. Finally, we show that a simple combination of these approaches improves upon the best results using any one approach.

This chapter is organized as follows. Section 8.1 describes the knowledge sources used in the models to be evaluated. Section 8.2 reviews the HMM approach that has been used in the previous chapters. Section 8.3 introduces the Maxent modeling approach and describes experimental results when using it on the SU detection task. Section 8.4 compares the CRF model with the HMM and Maxent models over several experimental conditions. Section 8.5 summarizes this chapter.

8.1 Knowledge Sources

In this section, we briefly summarize the knowledge sources that have been used previously in the HMM approach and will be used by the different modeling approaches compared in this chapter. Words and SU boundaries are mutually constrained via syntactic structure. Therefore, the word identities themselves (from automatic recognition or human transcriptions) constitute a primary knowledge source for the SU boundary detection task. We also make use of various automatic taggers that map the word sequence to other representations. The TnT tagger [106] is used to obtain POS tags. A TBL chunker [129] maps each word to an associated chunk tag, encoding chunk type and relative word position (beginning of a noun phrase, inside a verb phrase, etc.). The tagged versions of the word stream are provided to allow generalizations based on syntactic structure and to smooth out possibly undertrained word-based probability estimates. For the same reasons we also generate word class labels that are automatically induced from bigram word distributions [105]. Hidden event LMs based on these various tags were described in Chapter 6.

To model the prosodic structure of SU boundaries, prosodic features are extracted around each word boundary as described in Chapter 4. These are based on the acoustic alignments produced by a speech recognizer or forced alignments of the reference transcriptions when given. The features capture duration, pitch, and energy patterns associated with the word boundaries. A crucial aspect of many of these features is that they are highly correlated (e.g., some are derived from the same raw measurements via different normalizations), real-valued (not discrete), and possibly *undefined* (e.g., unvoiced speech regions have no pitch). These properties make prosodic features difficult to model directly together with the textual information in the HMM approach. Hence, a modular approach has been adopted: the information from the prosodic features is modeled separately by a decision tree classifier that outputs posterior probability estimates $P(e_i|f_i)$, where e_i is the boundary event after w_i, and f_i is the prosodic feature vector associated with the word boundary. Conveniently, this approach also permits us to include some non-prosodic features that are highly relevant to the task but are not otherwise represented in the generative HMM, such as whether a speaker (turn) change occurs at the location in question.

8.2 A Review of the HMM for SU Detection

Our baseline model, as described in Chapter 4, is a hidden Markov model (HMM). This model forms the basis of much of the prior work on sentence boundary detection in speech [12, 15–17]. We briefly review the HMM that is used for structural event detection in this section. This system has been evaluated in Chapters 5 through 7. In the HMM, the states of the model correspond to the word w_i and event label e_i that is associated with the word end boundary. The observations associated with the states are the words, as well as other prosodic features f_i. Figure 8.1 shows a graphical model representation of the variables involved. Note that the words appear in both the states and the observations, such that the word stream constrains the

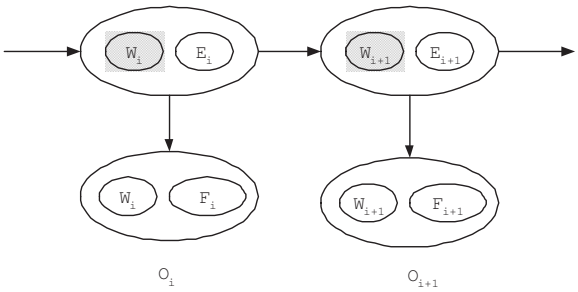

Fig. 8.1. The graphical model for the SU detection problem. Only one word-event pair is depicted in each state, but in a model based on N-grams the previous $N-1$ tokens would condition the transition to the next state. O are observations consisting of words W and prosodic features F, and E are structural events.

possible hidden states to matching words; the ambiguity in the task stems entirely from the choice of events.

For the HMM approach, standard algorithms are available to extract the most probable state (and thus event) sequence given a set of observations. Since our goal is to minimize the per-boundary error rate (and thus the NIST SU error rate), rather than finding the highest probability *sequence* of events, we identify the events with the highest posterior probability *individually* at each boundary i:

$$\hat{e}_i = \arg\max_{e_i} P(e_i|W, F) \tag{8.1}$$

where W and F are the words and features for the entire test sequence, respectively. The individual event posteriors are obtained by applying the forward-backward algorithm for HMMs [84].

Training of the HMM is supervised since event-labeled data is available. The state transition probabilities are estimated using a hidden event N-gram LM [83]. The LM is obtained with standard N-gram estimation methods from data that contains the

word-event pair tags in sequence: $w_1, e_1, w_2, \ldots e_{n-1}, w_n$. The resulting LM can then compute the required HMM transition probabilities as:[1]

$$\begin{aligned} P(w_i e_i | w_1 e_1 \ldots w_{i-1} e_{i-1}) = \\ P(w_i | w_1 e_1 \ldots w_{i-1} e_{i-1}) \times \\ P(e_i | w_1 e_1 \ldots w_{i-1} e_{i-1} w_i) \end{aligned}$$

The N-gram estimator maximizes the joint word-event pair sequence likelihood $P(W, E)$ on the training data (modulo smoothing), and does not guarantee that the correct event posteriors needed for classification according to Equation (8.1) are maximized.

The second set of HMM parameters are the observation likelihoods $P(f_i|e_i, w_i)$. Instead of training a likelihood model we make use of the prosodic classifier as described in Chapter 4. Observation likelihoods may be obtained as follows:

$$P(f_i|w_i, e_i) = \frac{P(e_i|f_i)}{P(e_i)} P(f_i) \tag{8.2}$$

where $p(e_i|f_i)$ is obtained from the decision tree estimation. As in Chapter 7, instead of a single decision tree, we use ensemble bagging to reduce the variance of the classifier and generate more reliable posterior probability estimation. In Chapters 4 and 7, we have described how to deal with the mismatch of class distributions between training and testing for decision tree learning.

The HMM modeling representation we adopt for the SU detection is different from the HMM that is used for the POS tagging problem or other sequence labeling tasks in NLP. For comparison, Figure 8.2 shows a graphical model for POS tagging, where the hidden states consist of POS tags, and the observations are the words. In the SU detection task, since there are only two classes (SU or not), we hypothesize that if they were used as states, the state sequence would not contain enough discriminative information to effectively decode the sequence. Hence, rather the words are added in the states to constrain the event sequence. Table 8.1 shows the SU detection results

[1]To utilize the word and event contexts of length greater than one, HMMs of order 2 or greater are used, or equivalently, use the entire word-event pair N-gram as the state.

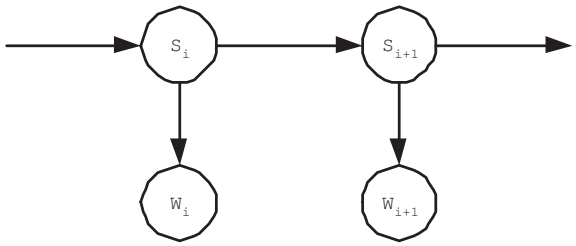

Fig. 8.2. The graphical model for the POS tagging problem. POS tags are the hidden states in this problem. S are POS tags, and W are words.

comparing models for which words appear in the states or not, on the CTS human transcription condition. Both approaches use trigram models that use only word information for sequence decoding (i.e., no prosodic information is used). Clearly there is a substantial performance degradation when word information is removed from the 'hidden' states; hence, the state configuration in Figure 8.1 is used in all of the remaining experiments.

Table 8.1
SU detection results (NIST error rate in %) for different state configurations using the trigram LM alone on the CTS reference condition. The insertion (INS), deletion (DEL), and total error rate are shown.

State membership	SU error rate (%)		
	INS	DEL	Total
event only	60.56	15.11	75.66
word and event	31.47	13.74	45.20

The HMM structure makes strong independence assumptions: (1) that features depend only on the current state (and in practice, only on the event label) and (2) that each word-event pair label depends only on the previous $N-1$ tokens. In return, we get a computationally efficient structure that allows information from the entire

sequence of words and prosodic features W, F to inform the posterior probabilities needed for classification, via the forward-backward algorithm.

More problematic in practice is the integration of multiple word-level features, such as POS tags and chunker output. Theoretically, all tags could simply be included in the hidden state representation to allow joint modeling of words, tags, and events. However, this would drastically increase the size of the state space, making robust model estimation with standard N-gram techniques difficult. A method that works well in practice is *linear interpolation*, whereby the conditional probability estimates of various models are simply averaged. Some improvement in SU boundary detection performance has been obtained by combining the word-based hidden event LM with the class-based LMs, as discussed in Chapter 6.

Similarly, we can interpolate LMs trained from different corpora. This is usually more effective than pooling the training data because it allows control over the contributions of the different sources. For example, we use LMs that are obtained from the extra, larger corpora in Chapter 6. Given a larger training corpus, it should get a larger weight; but given more imprecise labeling of SU boundaries, it should get a lower weight. By tuning the interpolation weight of the two LMs empirically (using held-out data), a good compromise between these two extremes can be found.

8.3 The Maxent Posterior Probability Model for SU Detection

The Maxent model [130] has been successfully applied to a variety of NLP tasks, such as POS tagging [131, 132], text categorization [133], chunking [134], machine translation [135], language modeling [102, 103, 136], named entity detection [137–139], and word sense disambiguation [140]. It has been found to perform similarly to other state-of-the-art approaches for these NLP tasks. Each model has taken a classification approach and has designed features accordingly. Most of features involve word context, lexical information about the word, and other word related information. Focus on feature design has led to superior performance, for example

[132] for POS tagging. We will show later that feature design is an important factor in Maxent performance for structural event detection.

8.3.1 Description of the Maxent Model

As discussed in Chapter 4, the SU detection problem is represented as a classification task. For a given word boundary, each observation consists of the context of the word and the corresponding prosodic features, and the task is to classify this boundary as an SU or not. The assumption here is that the observations $O_i = (F_i, C(w_i))$ are independent. Note that we have already pre-encoded some dependency into the feature set associated with each sample, e.g., using the word context $C(w_i)$. More contextual prosodic features could also be encoded in the observation O_i, not just using the features F_i associated with the word boundary.

For a classification problem, there could be different ways to set the model parameters θ, either to maximize the joint likelihood $P(E, O)$ or the conditional likelihood $P(E|O)$ over the entire training set, where E denotes the class labels. The posterior probability (likelihood) correlates more with the classification error rate metric than the maximum joint likelihood does, therefore the conditional likelihood is used as our objective function [141]:

$$CL(\theta, D) = P(E|O) = \prod_{e,o \in D} P(e|o) \tag{8.3}$$

where D is the training data set consisting of labeled examples (e, o), CL is conditional likelihood, and θ denotes the model. The conditional likelihood $p(e|o)$ is closely related to the individual event posterior probability used for classification, allowing this type of model to explicitly optimize discrimination of correct from incorrect labels.

The problem now is to estimate the conditional probability $p(e|o)$, where o is a vector of features, representing different knowledge sources, and e is the class label (SU or not). The Maxent model, which models things that are known and does not make assumptions about the unknown events, provides one possible solution. Con-

straints in the Maxent model are obtained from the training set, i.e., the empirical distribution is equal to the expected value of a feature functions g with respect to the model $p(y|x)$: $E_{\tilde{p}}[g_i] = E_p[g_i]$, i.e.:

$$\frac{1}{|D|}\sum_x g(x, y(x)) = \sum_x \tilde{p}(g) \sum_y \tilde{p}(x)p(y|x)g(x, y) \tag{8.4}$$

where x is the feature set associated with a sample, and y is its class label. The functions $g_i(x, y)$ are indicator functions corresponding to (complex) features defined over events, words, and prosodic features. For example, one such feature function for the SU detection task might be:

$$g(x, y) = \begin{cases} 1 : \text{if } w_i = \textit{uhhuh} \text{ and } y = \textit{SU} \\ 0 : \text{otherwise} \end{cases}$$

When there is no confusion, we sometimes call the predicate part in the indicator function 'features'. These are generally used as features by other machine learning approaches, such as decision trees.

The Maxent model finds a probability distribution that satisfies these constraints, and has the maximum conditional entropy:

$$H(p) = -\sum_x p(x)p(y|x) \log p(y|x) \tag{8.5}$$

The solution to this constrained optimization problem has the exponential form:

$$p(y|x) = \frac{1}{Z_\lambda(x)} exp(\sum_i \lambda_i g_i(x, y)) \tag{8.6}$$

where $Z_\lambda(x)$ is the normalization term:

$$Z_\lambda(x) = \sum_y exp(\sum_i \lambda_i g_i(x, y)) \tag{8.7}$$

To find the parameters λ, the log likelihood $\prod_i P(e_i|W, F)$ over the training data is maximized. In our experiments, the L-BFGS parameter estimation method is used, with Gaussian-prior smoothing [142] to avoid overfitting. Implementations for both methods are from the Maxent toolkit [143]. The training data for structural event

detection is very limited; therefore, smoothing is essential. The intuition behind the use of Gaussian priors is to force the parameters to be distributed according to a Gaussian distribution with mean μ and variance σ^2. This prior expectation penalizes parameters that drift away from their mean prior value (μ is 0 generally). When Gaussian smoothing is used, a penalty term is added to Equation (8.3) during the maximum likelihood estimation:

$$CL'(\lambda) = CL(\lambda) + \sum_{\lambda_i} log \frac{1}{\sqrt{(2 \times \pi \times \sigma^2)}} exp(\frac{-\lambda_i}{2 \times \sigma^2}) \tag{8.8}$$

8.3.2 Features Used

Even though Maxent gives us the freedom to use features that are overlapping or otherwise dependent, we still must choose a subset of features that are informative and parsimonious in order to obtain good generality and robust parameter estimates. We included all features that correspond to information available to our HMM approach, which are summarized below. Those features that are triggered only once in the training set are then eliminated to improve robustness and to avoid overfitting the model. More discussion on feature selection will appear in the next section.

- *Word N-grams.* Combinations of preceding and following words are used to encode the word context of the event, e.g., $<w_i>$, $<w_{i+1}>$, $<w_i, w_{i+1}>$, $<w_{i-1}, w_i>$, $<w_{i-2}, w_{i-1}, w_i>$, and $<w_i, w_{i+1}, w_{i+2}>$, where w_i refers to the word before the boundary of interest.

- *POS N-grams.* POS tags (see Section 6.2) are the same as used for the HMM. The features capturing POS context are similar to those based on word tokens.

- *Chunker tags.* These are used similarly to POS and word features, except tags encoding chunk type (NP, VP, etc.) and word position within the chunk (beginning versus inside) are used. Chunker features are only used for the BN data. These syntactic chunk tags are generated from a TBL chunker as described in Section 6.2.

- *Word classes.* These are similar to N-gram patterns but over automatically induced classes that are obtained in the same way as described in Section 6.2.

- *Turn flags.* Since speaker change often marks an SU boundary, this binary feature is used. Note that in the HMM this feature is grouped with the prosodic features and handled by the decision tree since it is not easy to capture this information using the hidden event LM in the HMM; whereas, in the Maxent approach, it can be used separately as a feature.

- *Prosody.* Even though it is possible to include prosodic features in the Maxent framework, designing a method to encode the many various prosodic features used by the decision trees is not straightforward. Therefore we decided to use the decision tree classifiers to generate the posterior probabilities $p(e_i|f_i)$. Since it is convenient to use binary features in the Maxent classifier, the prosodic posteriors are encoded into several binary features via thresholding. Equation (8.6) allows each feature in a Maxent model to have a monotonic effect on the final probability (raising or lowering it by a constant factor $e^{\lambda_k g_k}$). This suggests encoding the decision tree posteriors in a cumulative fashion through a series of binary features, for example, $p > 0.1$, $p > 0.3$, $p > 0.5$, $p > 0.7$, $p > 0.9$. These thresholds are heuristically chosen. This representation has the advantage that it is more robust to a possible mismatch between the posterior probability in training and test sets, since small changes in the posterior value affect at most one feature.

  In order to obtain the posterior probability from the prosody model for the training samples, a cross-validation approach is used. If we were to train the decision trees from the training set and generate the posterior probability on the same set using these trees, then the probabilities would probably be biased. Hence, a 10-fold cross-validation is used: the training set is split into 10 subsets and the trees trained from 9 sets are used to generate the posterior probabilities for the one that is left out. In this way, each sample in the training set is

assigned a probability which is binned for use in the Maxent model. In testing, the trees that are trained from the entire training set are used to estimate probabilities.

Although that the Maxent framework does allow the use of real-valued feature functions, our preliminary experiments have shown no gain compared to the use of binary features constructed as described above. Results using continuous features in the Maxent model are shown in Section 8.3.4.

- *Auxiliary LM.* As mentioned earlier, additional text-only language model training data is often available. In the HMM model, we incorporated auxiliary LMs by interpolation, which is not possible here since there is no LM *per se*, but rather N-gram features. However, the same trick can be used as is used for the prosodic features. A word-only HMM is used to estimate posterior event probabilities based on the auxiliary LM, and these posteriors are then thresholded to yield binary features. The auxiliary LM is applied to both the training and the test set of the RT-03 training and test data to generate the posterior probability estimations for CTS and BN respectively.

- *Combined features.* To date we have not fully investigated compound features that combine different knowledge sources in order to model their interaction explicitly. Only a limited set of such features is included, for example, a combination of the decision tree hypothesis and POS contexts. The reason we choose to use POS tags together with the decision trees' hypotheses is to limit the number of parameters (since there are only limited number of POS tags), while attempting to combine some grammatical constraints with prosodic information.

8.3.3 Comparisons of the Maxent and HMM Approaches

HMM training does not directly maximize the posterior probabilities of the correct labels, resulting in mismatch between training and the use of the model as a classifier. A second problem with HMMs is that the underlying N-gram sequence model does not cope well with multiple representations (features) of the word sequence (words, POS, etc.) short of building a joint model involving all variables. These problems can be well addressed by the Maxent model. A Maxent model directly estimates the posterior boundary label probabilities $P(e_i|W, F)$, rather than maximizing the joint likelihood of the observation and the state sequence. The Maxent model allows correlated features to apply simultaneously, and therefore gives greater freedom for combining knowledge. Another desirable characteristic of the Maxent models is that they do not split the data recursively to condition their probability estimates, which makes them more robust than decision trees when training data is limited. Hence it is possible to include prosodic features directly into the Maxent framework, instead of modeling them separately using decision trees.

The HMM and Maxent differ regarding the training objective function (joint likelihood versus conditional likelihood) and with respect to the handling of dependent word features (model interpolation versus integrated modeling via Maxent). On both counts the Maxent classifier should be superior to the HMM. However, the Maxent approach also has some theoretical disadvantages compared to the HMM by design. One obvious shortcoming of the Maxent approach is that some information is lost by the thresholding that converts posterior probabilities from the prosodic model and the auxiliary LM into binary features.[2] A more qualitative limitation of the Maxent model is that it only uses local evidence, namely, the surrounding word context and the local prosodic features. Based on this, the Maxent model resembles the conditional probability model at the individual HMM states. The HMM as a

[2]This is not a drawback of the Maxent classifier itself, rather a design used for this SU detection task.

whole, however, through the forward-backward procedure, propagates evidence from all parts of the observation sequence to any given decision point.

8.3.4 Results and Discussion for the Maxent SU Model

Experiments comparing the Maxent and HMM approaches were conducted for the SU detection task on the BN and CTS corpora. Training and test data are the same as those used in the experiments in Chapters 5. System performance is evaluated using the official NIST evaluation tools.

In our experiments, we compare how the two approaches perform individually and in combination. The combined classifier is obtained by simply averaging the posterior estimates from the two models, and then picking the event type with the highest probability at each position. Other experimental factors are also investigated, such as the impact of the speech recognition errors, the impact of genre, and the contribution of textual versus prosodic information in each model.

Experimental Results

Table 8.2 shows SU detection results for BN and CTS, on both the reference transcriptions and speech recognition output conditions, using the HMM and the Maxent approach individually and in combination. The Maxent approach slightly outperforms the HMM approach when evaluating on the reference transcriptions, and the combination of the two approaches achieves the best performance for all the tasks (significant at $p < 0.05$ using the sign test on the REF condition, mixed results on the STT condition).

We observe in Table 8.2 that there is a large increase in error rate when evaluating on speech recognition output. This replicates findings from previous chapters. The Maxent system degrades more than the HMM when recognition output is used. As can be seen from Table 8.2, the Maxent outperforms the HMM when the reference transcription is used; however, in the STT condition, the Maxent yields comparable

Table 8.2
SU detection results (NIST error rate in %) using the Maxent and the HMM approaches individually and in combination on BN and CTS, on reference transcriptions (REF) and recognition output (STT).

		HMM	Maxent	Combined
BN	REF	48.72	48.61	46.79
	STT	55.37	56.51	54.35
CTS	REF	31.51	30.66	29.30
	STT	42.97	43.02	41.88

or slightly worse performance than the HMM. This makes sense since most of the improvement of the Maxent model comes from better lexical feature modeling. But these are exactly the features that deteriorate most with recognizer errors. On the other hand, prosodic information is more robust in face of recognition errors and yet it is not fully utilized in the Maxent approach.

Table 8.3 shows the deletion and insertion error rates for the HMM and the Maxent approaches on the reference condition. Due to the reduced dependence on the prosody model, the errors made by the Maxent approach are different from the HMM approach. There are more deletion errors and fewer insertion errors, since the prosody model tends to overgenerate SU hypotheses. We consistently observe a decrease in deletion rate and an increase in insertion error rate when the LM is combined with prosody model (compared to using the LM alone). The different error patterns suggest that we can effectively combine the system output from the two approaches, which is confirmed in Table 8.2, where the combination consistently yields the best performance (significantly better than the HMM alone at $p < 0.05$ using sign test).

Table 8.4 shows SU detection results for the two approaches, using textual information only, as well as in combination with the prosody model. We focus on the

Table 8.3
Deletion, insertion, and total error rate (NIST error rate in %) of the HMM and Maxent approaches on reference transcriptions of BN and CTS.

		DEL	INS	Total
BN	HMM	28.48	20.24	48.72
	Maxent	32.06	16.54	48.61
CTS	HMM	17.19	14.32	31.51
	Maxent	19.97	10.69	30.66

results for the reference transcription condition. The Maxent achieves a lower error rate than the HMM when using only the textual features; however, when prosodic information is incorporated, the gain diminished. The superior results for text-only classification are consistent with the Maxent model's ability to combine overlapping word-level features in a principled way. In contrast, the HMM approach linearly interpolates various LMs (see Chapter 6) at the LM level or posterior probability level. However, the HMM largely catches up once prosodic information is added. This can be attributed to the direct use of prosodic posterior probabilities in the HMM, as well as the fact that in the HMM, each boundary decision is affected by prosodic information throughout the whole sequence; whereas, the Maxent model uses only the prosodic features at the boundary to be classified.

Notice also from Table 8.2 that the Maxent approach yields more gain over the HMM on CTS than on BN (e.g., for the REF condition on both corpora). One possible reason for this is that there is more training data and thus less of a sparse data problem for CTS. Another possible reason is that the prosody model contributes more on BN than on CTS (a pattern observed also in Chapter 5), and the role of that component is lower in the Maxent approach. As can be seen from Table 8.4, when using textual information only, the gain (relative error rate reduction) from the Maxent over the HMM is slightly less on BN than on CTS.

Table 8.4
SU detection results (NIST error rate in%) using different knowledge sources on BN and CTS, evaluated on the reference transcription.

		BN	CTS
HMM	Textual	67.48	38.92
	Textual + prosody	48.72	31.51
Maxent	Textual	63.56	36.32
	Textual + prosody	48.61	30.66

Additional Investigations

The Maxent model makes better use of lexical features, but does not use the prosodic information as effectively partly due to accumulatively binning the posterior probabilities from the prosody model. However, the framework of the Maxent itself is not restricted to binary features; if the features have a continuous value, that value can be used when weighting the features. Hence, we investigate whether preserving the posterior probabilities from the prosody model improves the sensitivity of the Maxent model to prosodic information. Table 8.5 shows the results of the Maxent model using the same features as in our earlier experiment (the features listed in Section 8.3.2), with the exception that the posterior probabilities from the decision trees are now real-valued features. This experiment is performed using the reference transcriptions from the CTS data. We consider using the posterior probability from the prosody model as is and also using its log value in the Maxent framework.

Clearly there is no improvement when the posterior probability is represented as a real-valued feature. The degradation is not substantial when the real-valued posterior probabilities are used as is; however, if the log value of the probabilities is used, there is a significant degradation in performance. A potential advantage of using continuous features could result from using the confidence scores generated by

Table 8.5
Comparison of using the posterior probabilities from the prosody model as binary features versus continuous valued features in the Maxent approach for SU detection in CTS reference transcription condition.

Posterior probabilities		SU error rate (%)
binary	cumulative binned probability	30.66
continuous	posteriors	30.82
	log posteriors	31.77

the speech recognizer, so that each textual feature will be weighted based on their confidence measures on the STT test condition.

In another study, feature selection or feature pruning is investigated in the Maxent approach. Feature selection is important, since some features can be noisy and thus hurt performance. In addition, overfitting may result when features do not generalize well to the test set. Removing irrelevant features can generate more accurate predictions and a more compact model. Various feature selection algorithms for the Maxent models have been investigated, e.g., [130]. We consider some of feature selection metrics that have been previously studied for text classification [144].

- **Information gain (IG):** This is widely used in machine learning and data mining as a measure of association. It is based on the definition of entropy and conditional entropy:

$$IG(Y|x) = H(Y) - H(Y|x) \tag{8.9}$$

where $H(Y)$ is the entropy:

$$H(Y) = \sum_{y} p(y) \log p(y) \tag{8.10}$$

$H(Y|x)$ is the conditional entropy:

$$H(Y|x) = \sum_{y} p(y|x) \log p(y|x) \tag{8.11}$$

x is a feature, and Y is a random variable representing the class membership. Information gain measures the number of bits saved when transferring Y because of the presence of x. If x does not provide any information about Y, then the IG value is 0, otherwise it is positive.

Given a training set, the information gain that each feature provides for a two-way SU classification task is calculated as follows:

$$\begin{aligned} H(Y) &= -(p(SU) \times \log p(SU) + p(\overline{SU}) \times \log p(\overline{SU})) \\ H(Y|x) &= -(p(SU|x) \times \log p(SU|x) \\ &\quad +p(\overline{SU}|x) \times \log p(\overline{SU}|x)) \end{aligned} \tag{8.12}$$

Table 8.6 shows some of the N-gram features with the highest IG weights for the SU detection task. Clearly, the selected words are good signals for SU boundaries, most of which are sentence initial and final words.

- **Mutual information (MI):** The mutual information[3] between a feature f and a class c is defined as:

$$\begin{aligned} MI(f,c) &= \log p(f|c) - logp(f) \\ &= logp(c|f) - logp(c) \end{aligned} \tag{8.13}$$

and the average of $MI(f,c)$ over all the classes is the mutual information between the feature x and the class variable Y:

$$\begin{aligned} MI(x,Y) &= \sum_{c_i} p(c_i) MI(f,c_i) \\ &= \sum_{c} p(c)(\log p(c|x) - \log p(c)) \end{aligned} \tag{8.14}$$

[3]Sometimes IG and MI are not well defined and are used interchangeably in the literature; hence, we indicated how we use these terms.

Table 8.6
Some of the N-gram features with the highest IG weights for the CTS SU detection task.

Feature	**IG weight**
current word is YEAH	0.026768
next word is YEAH	0.025801
current word is UHHUH	0.019321
next word is UHHUH	0.019032
next word is AND	0.016337
next word is BUT	0.010515
next word is OH	0.008910
current word is OH	0.007039
next word is WELL	0.005783
current word is RIGHT	0.005663
current and next words YEAH YEAH	0.005369
next word is SO	0.004947
current word is I	0.004632
current word is THE	0.004045

Mutual information represents the reduction in uncertainty about Y due to the knowledge of x. MI can be easily computed from the training set. Note that the mutual information is equal to:

$$H(Y) + \sum_{c} p(c) \log p(c|f) \tag{8.15}$$

This is similar to, but different from the information gain, which is:

$$H(Y) + \sum_{c} p(c|f) \log p(c|f) \tag{8.16}$$

- **Chi-square statistics:** Chi-square measures the independence between a feature and a class by using a Chi distribution. For a two by two contingency table as shown in Table 8.7, the Chi statistic is defined as:

$$Chi = \frac{N \times (AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{8.17}$$

For each feature, we find the Chi statistic results from the Chi distribution using a degree of freedom of 1. The Chi-square statistic may not be reliable when the event frequency is low [145].

Table 8.7
Notation for a 2 × 2 contingency table used in Chi-square statistics.

	feature occurs	feature does not occur
class occurs	A	B
class does not occur	C	D

We compute the weight for each feature in the training set using the metrics described above and sort the features based on their weights. The N features that have the highest weights are preserved and the others are pruned. Table 8.8 shows the results when different numbers of the features are preserved using the weights based on each of the three metrics. Experiments were conducted on the reference transcriptions of the CTS data. Our experiments show that preserving more features yields a consistent gain. Using all the features outperforms the pruned feature sets (the difference is significant at $p < 0.05$). This is not very surprising since Gaussian prior smoothing can make the model parameter estimates more robust, mitigating issues of noisy features or overfitting. For each feature selection metric, the difference between using 1 million and 2 million features is significant (at $p < 0.01$). Mutual information outperforms the other two feature selection algorithms. There is no significant difference between using information gain and the Chi-square metrics for feature selection.

Table 8.8
SU detection results (NIST error rate in %) using different feature selection metrics and different pruning thresholds (number of the preserved features), for the CTS REF condition.

Number of preserved features	SU error rate (%)		
	IG	MI	CHI
0.5 million	33.46	32.03	32.71
1 million	33.07	31.88	32.96
2 million	32.24	31.45	32.21
all the features (5.1 million)	30.66		

8.4 The Conditional Random Field (CRF) Model for SU Detection

A simple combination of the Maxent and HMM was found to improve upon the performance of either model alone (as shown in Table 8.2). This is likely due to the complementary strengths and weaknesses of the two models. An HMM is a generative model, yet it is able to model the sequence via the forward-backward algorithm. The Maxent approach is a discriminative model; however, it attempts to make decisions locally, without using sequential information. A conditional random field (CRF) model [146] combines the benefits of these two approaches. Like the Maxent, a CRF can accommodate many correlated features and can be trained in a discriminative way. Like an HMM, a CRF uses sequence decoding that is globally optimal. Hence, we will compare the performance of the CRF model to both the HMM and Maxent approaches on the SU detection task.

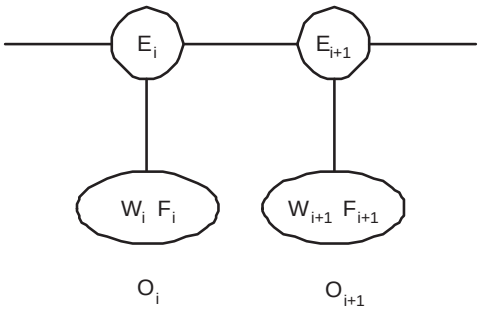

Fig. 8.3. The graphical representation of a CRF for the sentence boundary detection problem. E represents the state tags (i.e., SU boundary or not), while W and F are word and prosodic features respectively. O are observations consisting of W and F.

8.4.1 Description of the CRF Model

A CRF is a random field that is globally conditioned on an observation sequence X. CRFs have been successfully used for a variety of text processing tasks, such as parsing [147], named entity recognition [148], and information extraction [149].

Figure 8.3 depicts a graphical representation of this modeling approach for the sequence labeling task. The CRF is an undirected graph, in which the states of the model correspond to event labels E_i, and the observations X_i associated with the states are the words W_i, as well as other prosodic features F_i. The most likely sequence $\hat{E}$ for the given input sequence (observations) X is:

$$\hat{E} = \arg\max_{E} \frac{exp(\lambda \times G(E, X))}{Z_\lambda(X)} \tag{8.18}$$

where the function G is a potential function over the events and the observations, and Z_λ is the normalization term

$$Z_\lambda(X) = \sum_{E} exp(\lambda \times G(E, X)) \tag{8.19}$$

The CRF model is trained to maximize the conditional log-likelihood of a given training set. Like the Maxent model, this is closely related to the performance metric. The most likely sequence is found using the Viterbi algorithm.[4]

The Mallet package [150] is used to implement the CRF model. To avoid overfitting, a Gaussian prior is employed with a mean of zero on the parameters [142], similar to what is used for training Maxent models. The features used in the CRF SU detection model are the same as those used in the Maxent approach. The CRF takes longer to train than do the HMM and Maxent models, especially when the number of features becomes large. The HMM requires less time for training than the other two models.

8.4.2 Comparisons of CRF and Other Models

A CRF, like the Maxent, differs from an HMM with respect to its training objective function (joint versus conditional likelihood) and its handling of dependent word features. HMM training does not maximize the posterior probabilities of the correct labels; whereas, the CRF directly estimates posterior boundary label probabilities $P(E|W,F)$. The underlying N-gram sequence model of an HMM does not cope well with the overlapping representations of the word sequence; however, the CRF model supports simultaneous correlated features and therefore allows us to easily incorporate a variety of knowledge sources. A CRF, like the HMM, differs from the Maxent method with respect to its ability to model sequence information. The primary advantage of the CRF over the Maxent approach is that the model is optimized globally over the entire sequence; whereas, the Maxent model uses only local evidence. The CRF is essentially a Maxent model at the sequence level, i.e., the entire sequence is treated as one sample for the posterior probability estimation in the Maxent framework, but it is implemented using the efficient Viterbi algorithm.

[4]The forward-backward algorithm would likely be better here, but it is not implemented in the current software used [150].

In addition, the CRF differs from a maximum entropy conditional Markov model (CMM). In a CMM, a single function $p(s|s', o)$ represents the probability of the current state s given the previous state s' and the current observation o. However, this approach has a known problem, called *label bias problem*, due to the per-state normalization of the transition scores. This causes problems when there are fewer outgoing states. For example in the extreme case, if there is only a single state, then the observations are equivalently ignored. The CRF approach addresses this label bias problem of the CMM by employing sequence modeling.

Figure 8.4 compares the graphical models of the HMM, CMM, and CRF approaches. The HMM is a generative model, which models the joint distribution of $P(S, O)$ and in which the state is dependent only on the previous state (or a limited set of previous states if higher orders of HMMs are used). In the CMM, the state depends on the previous state and the observation. In the CRF sequence modeling approach, a single exponential form is used for the probability of the state sequence given the entire observation, rather than the per-state exponential form in the CMM approach.

8.4.3 Results and Discussion for the CRF SU Model

The features used for the CRF are the same as those described in Section 8.3.2 for the Maxent model, which also happen to be the knowledge sources for the HMM (but with different representations). Keeping the knowledge sources consistent across the models enables us to focus on the comparison of the effectiveness of the modeling approaches. However it is worth noting that it is possible that different modeling approaches (e.g., CRF) could make use of new features, which has not been explored yet. For this investigation, we compare the CRF, HMM, and Maxent models to one another, as well as to a voting-based combination of the three.

SU detection results using the CRF, HMM, and Maxent approaches individually, using the reference transcriptions or recognition output for CTS and BN are shown

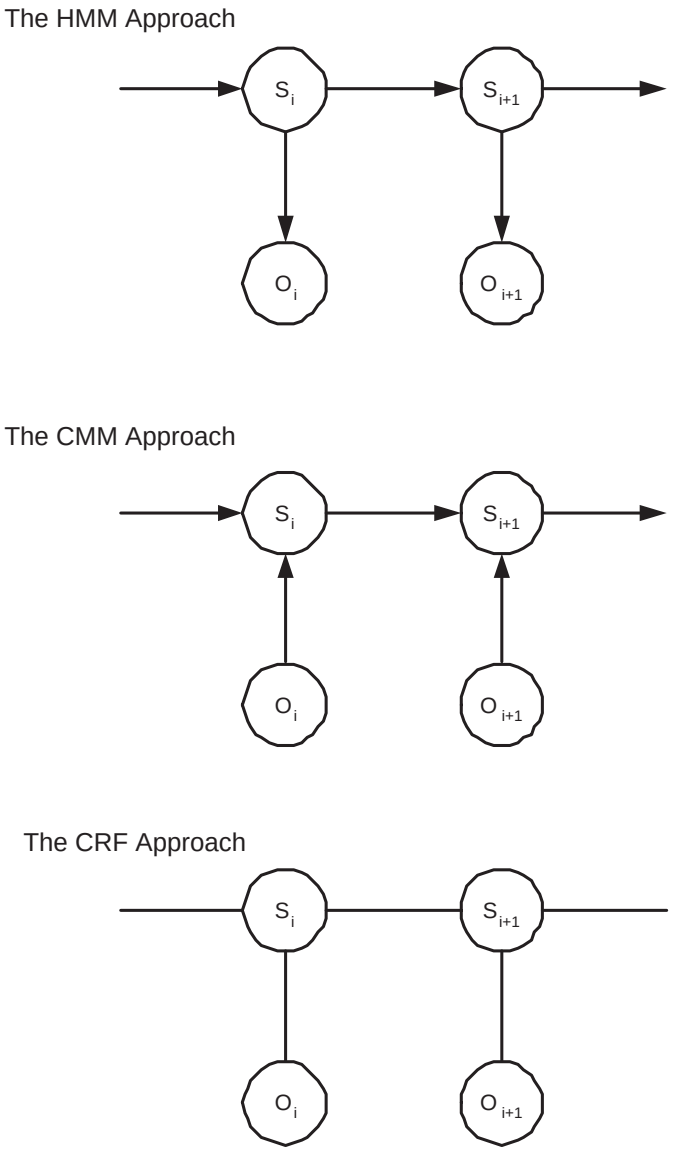

Fig. 8.4. The graphical model representations of the HMM, CMM, and CRF approaches. O are observations, and S are events (or tags).

in Tables 8.9, along with a combination of the three modeling approaches using a majority vote. Results are reported using the NIST SU error rate.

As can be seen from the table, the CRF is superior to both the HMM and Maxent across all conditions (the differences are significant at $p < 0.05$) except for the BN

Table 8.9
SU detection results (NIST error rate in %) using the HMM, Maxent, and CRF approaches individually and in combination on BN and CTS, on reference transcriptions (REF) and recognition output (STT). The combination of the three approaches is obtained via a majority vote.

		HMM	Maxent	CRF	Majority vote
BN	REF	48.72	48.61	47.92	46.28
	STT	55.37	56.51	55.37	54.29
CTS	REF	31.51	30.66	29.47	29.30
	STT	42.97	43.02	42.00	41.88

STT condition. The combination of the three approaches is superior to any model alone. Previously, we found that the Maxent and HMM posteriors combine well (see Table 8.2). The toolkit that is used for the implementation of the CRF does not provide a posterior probability for a sequence; therefore, we were unable to combine the system output via posterior probability interpolation, which we would expect to yield a stronger performance gain. We observe from Table 8.9 the CRF has a larger increase in error rate when evaluating on speech recognition output compared to the HMM, suggesting that the CRF suffers more from the recognition errors.

The CRF yields relatively less gain over the HMM on BN than on CTS. One possible reason for this difference is that there is more training data for the CTS task, and both the CRF and Maxent approaches require a relatively larger training set than the HMM. It is also possible that there is more effective sequence information in CTS due to the different speaking styles in CTS and BN. Since the CRF model is only a first-order Markov model, it is likely that it is better at modeling shorter SUs on CTS than the longer SUs on BN.

Similarly to the comparisons for the HMM and Maxent approaches, we investigate the impact of different knowledge sources. Table 8.10 and Table 8.11 show SU

Table 8.10
CTS SU detection results (NIST error rate in %) using the HMM, Maxent, and CRF individually, using different knowledge sources. Note that the 'all features' condition uses all the knowledge sources described in Section 8.3.2.

CTS				
		HMM	Maxent	CRF
REF	word N-gram	42.02	43.70	37.71
	word N-gram + prosody	33.72	35.09	30.88
	all features	31.51	30.66	29.47
STT	word N-gram	53.25	53.92	50.20
	+ word N-gram prosody	44.93	45.50	43.12
	all features	43.05	43.02	42.00

detection results for CTS and BN respectively, when different knowledge sources are used: word N-gram only, word N-gram and prosodic information, and using all the features listed in Section 8.3.2.

We observe from the tables that when only the word N-gram information is used, the gain of the CRF over the HMM or Maxent is the greatest, with the differences between the models diminishing as more features are added. This may be due to the impact of the sparse data problem on the CRF or simply due to the fact that differences between modeling approaches are less when features become stronger, that is, strong features compensate for weaknesses in the models. Notice that on CTS, with fewer knowledge sources (e.g., using only word N-gram and prosodic information) the CRF is able to achieve a performance similar to or better than other methods using all the knowledges sources. This may be useful when feature extraction is computationally expensive. Looking at the results when only word N-gram information is used, we observe the effect of word errors on each of the models. The SU detection error rate increases more in the STT condition for the CRF model

Table 8.11
BN SU detection results (NIST error rate %) using the HMM, Maxent, and CRF individually, using different knowledge sources.

BN				
		HMM	Maxent	CRF
REF	word N-gram	80.44	81.30	74.99
	word N-gram + prosody	59.81	59.69	54.92
	all features	48.72	48.61	47.92
STT	word N-gram	84.71	86.13	80.50
	word N-gram + prosody	64.58	63.16	59.52
	all features	55.37	56.51	55.37

than for the other models, suggesting that the discriminative CRF model suffers more from the mismatch between the training (which uses the reference transcription) and the test condition (in which features are obtained from potentially erroneous words).

8.5 Chapter Summary

Three different approaches have been described for modeling and integrating diverse knowledge sources for SU detection: a state-of-the-art approach based on HMMs, an alternative approach based on posterior probability estimation via the Maxent method, and a CRF sequence decoding approach. To achieve competitive performance using the Maxent and CRF models, we devised and evaluated a cumulative binary coding scheme to map posterior estimates from auxiliary submodels (prosody model and auxiliary LM) into features.

The HMM and Maxent approaches have complementary strengths and weaknesses that are reflected in the results, consistent with the findings for text-based NLP tasks [141]. The Maxent model is a discriminative approach that yields much

better accuracy than the HMM with lexical information, but a smaller win after combination with prosodic features possibly because of poorer prosodic feature modeling currently used for this approach. The HMM is a generative approach that in our modeling can currently make more effective use of prosodic information and thus degrades less with erroneous word recognition. An interpolation of posterior probabilities from the two systems achieves a 2-7% relative error reduction compared to the baseline (significant at $p < 0.05$ for the reference transcription condition). The results were consistent for two different genres of speech. Additionally we observe that Maxent is affected by the recognition errors more than the HMM approach, due to its heavy reliance on the textual information and lighter use of the prosodic information. Results also show that feature selection is not an important issue for the Maxent approach and pruning features degrades performance.

Our investigations have shown that a discriminatively trained CRF model is a competitive approach for the SU detection task. The CRF combines advantages of the generative HMM approach and the conditional Maxent approach, being discriminatively trained and able to model the entire sequence. It outperforms the HMM and Maxent approaches consistently across various testing conditions. We also find that as more knowledge sources are used, the differences among the modeling approaches decrease. A simple combination of the three modeling approaches (via majority vote) has proven superior to any single model.

Future useful work in this area would include developing more features that combine multiple knowledge sources, incorporating prosodic features directly in the Maxent and CRF approaches, generating posterior probabilities for an event in the CRF approach, as well as investigating approaches that model recognition uncertainty in order to mitigate the effects of word errors.

9. SYSTEM FOR RT-04

In this chapter, we describe some new methods that have been used in the latest NIST RT-04 evaluation and also verify that the approaches that have been investigated in earlier chapters hold for a new data set. This chapter is organized as follows. Section 9.1 describes the task and the data used in the RT-04 evaluation. Section 9.2 discusses the system performance for the SU task. Section 9.3 introduces the method used for SU subtype detection. Section 9.4 describes some new approaches for edit word detection and compares them with the previous approach described in Section 5.2.3. Section 9.5 summarizes this chapter.

9.1 RT-04 Tasks and Data

The structural event detection tasks in RT-04 are like those in the RT-03 evaluation (as described in Chapter 3), except that for the SU detection, not only are the positions of the SU boundaries generated, but also the subtype of each SU. The RT-04 data is annotated using the LDC V6 annotation guideline [73], which differs from the V5 guideline [71] that was used to annotate the RT-03 data (Table 3.2). For BN, there are some minor changes (e.g., more conventions are introduced for identifying sign offs and subordinating conjunctions), resulting in no significant differences between the annotations for the data of RT-03 and RT-04. Given this and the fact that the data sparsity problem is more severe for BN, we combine the RT-03 and RT-04 BN data, in order to increase the training data size. In contrast, the differences between the two guidelines for CTS are significant enough that only the RT-04 training data is used. The V6 annotation guideline uses open classes for filler words and backchannels, rather than a pre-defined word list as was used in the V5 guideline [71]. Additionally, more conventions and rules are introduced, for

example, in order to distinguish incomplete SUs and restart edit disfluencies. Given these differences, it is not straightforward to automatically map the RT-03 training data to the V6 annotation guideline. Therefore the RT-03 CTS data is not used. Table 9.1 describes the training and testing data used in the RT-04 NIST evaluation. The WER on the STT output used for the structural event detection tasks is also reported in the table. The CTS STT output is a combined IBM and SRI system; and the BN STT output is from the combination of all the RT-04 EARS systems.

Table 9.1
Data description for CTS and BN used in the RT-04 NIST evaluation. BN training data is the combined RT-03 and RT-04 data. CTS contains only the RT-04 training data.

	BN	CTS
Training	merged RT-03 and RT-04 data 40 hours	RT-04 data only 40 hours
Test	6 hours (12 shows)	3 hours (36 conversations)
STT WER (%)	11.7	18.6

9.2 System Performance for SU Boundary Detection

In this section, we discuss results on the SU boundary detection task using the new RT-04 data. Table 9.2 shows the SU boundary detection results. We compare the baseline system described in Chapter 5 with systems incorporating improvements investigated in Chapters 6 through 8. The baseline system uses an HMM that combines the word-based hidden event 4-gram LM trained from the data shown in Table 9.1 and the prosody model trained from a downsampled (DS) training set.

In Table 9.2, we observe significant improvements over the HMM baseline system by applying bagging to the prosody model and incorporating additional textual knowledge sources in the HMM. The Maxent and CRF generally outperform the

Table 9.2
SU boundary detection results (NIST SU error rate %) on the RT-04 evaluation data. The 'combination' is the majority vote of the Maxent, CRF, and the improved HMM approaches. 'DS' denotes a downsampled training set.

Models	SU Boundary Error Rate (%)			
	CTS		BN	
	REF	STT	REF	STT
Baseline HMM: word LM + prosody (DS)	33.00	44.58	60.97	71.35
Improved HMM: all LMs + prosody (bagging on DS)	28.66	40.47	51.76	63.34
Maxent	27.59	40.27	49.36	60.80
CRF	26.27	40.27	49.58	60.50
Combination	26.21	39.18	47.94	59.57

improved HMM, with a gain even greater than that found for the RT-03 data set in Chapter 8 for the BN data set. Also notice from the table that the Maxent and CRF approaches do not degrade as much on the STT condition compared to the results in Chapter 8. This may be due to better recognition accuracy and a larger training set (especially for BN). The combination results in the table are from the majority vote of the improved HMM, Maxent, and CRF approaches. As explained in Chapter 8, we do not currently have access to the posterior probabilities from the CRF tool we are using; therefore, a voting scheme is used for system combination. One would expect that other combination methods would improve upon this result.

9.3 SU/SU-Subtype Detection

In the SU/SU-subtype detection task, the end of an SU needs to be detected, and also the subtype for that SU. For this task, a two-step approach is adopted. First the SU boundary is detected using an HMM, Maxent, CRF, or some combination of these, then for each system hypothesized SU boundary, a classifier is used to determine its subtype. The reason we utilize a two-pass approach, rather than using the boundary detection approach with a 5-way classification (four SU subtypes plus non-SU) is to more easily incorporate knowledge about boundary locations (SU and SU initial words) into the subtype decisions. This kind of information would be difficult to directly incorporate in a one-pass 5-way classification approach. Hence, after the SU boundary is detected, a second-pass is used to determine the boundary type.

A Maxent classifier is used for SU subtype detection because of the ease of incorporating various features such as sentence initial cue words. These features would be hard to model using the current generative HMM. Features used include SU initial words (after optional filler words), SU final words, whether there is a turn change at the current and the previous SU boundaries, the length of the SU, and the binned posterior probabilities from a prosody model that does four-way SU subtype classification at a given SU boundary.

Table 9.3 shows the percentage of the four SU subtypes in CTS and BN data. For BN, statement is much more frequent than the other subtypes; whereas, for CTS the four types are more balanced, although statement remains the majority class. The highly skewed distribution of the subtypes on BN suggests that reasonably good performance can be achieved by hypothesizing statement SU for every SU boundary; hence the SU subtype detection task is only investigated for CTS.

SU subtype detection results are shown for CTS in Table 9.4. The first step boundary detection uses the majority vote of the HMM, Maxent, and CRF approaches (same as in the last row of Table 9.2). We report the boundary detection

Table 9.3
Percentage of SU subtypes for CTS and BN.

	statement	backchannel	question	incomplete
BN	94.23	0.86	4.37	0.53
CTS	62.05	26.80	5.11	6.05

Table 9.4
SU/SU-subtype detection results (%) on RT-04 CTS evaluation data. Results are reported using NIST SU boundary error rate, substitution error rate, and the subtype classification error rate (CER).

CTS			
	Boundary error	Substitution error	Subtype CER
REF	26.21	10.59	12.85
STT	39.18	10.07	13.29

error and the substitution error, both of which are from the NIST SU scoring tools. These errors are measured against the total number of reference SUs. Also shown in Table 9.4 is the SU subtype classification error rate (CER), defined as the percentage of the incorrectly labeled SU boundaries. The denominator in CER metric for SU subtype detection is the total number of the correct system hypothesized SUs, rather than all the reference SUs. This metric better represents the classifier's (the Maxent classifier here) performance, factoring out the boundaries that are missed in the system hypothesized SUs. Interestingly, the substitution errors or subtype classification error rates generally are not much affected by the STT errors or the SU hypothesis errors. Table 9.5 shows SU subtype classification performance in a confusion matrix for the reference transcription condition. As can be observed, there is a relatively larger percentage of misclassified boundaries for incomplete and question subtypes,

which are less frequent than the other subtypes (see Table 9.3), and they may also be more difficult to discern from the other types.

Table 9.5
SU subtype detection results (in confusion matrix) on CTS human transcription condition. Each cell shows the count and percentage (%) of a reference subtype (row) that is hypothesized as the subtype shown in the column.

		System hypothesis			
		backchannel	incomplete	question	statement
Reference	backchannel	1199 (84.98)	11 (0.78)	4 (0.28)	197 (13.96)
	incomplete	6 (2.05)	167 (57.19)	10 (3.42)	109 (37.33)
	question	2 (0.78)	2 (0.78)	158 (61.48)	95 (36.87)
	statement	56 (2.62)	30 (1.40)	25 (1.17)	2030 (94.82)

Note that in testing, features are extracted based on the system hypothesized SU boundaries; therefore, the starting point for an SU may be wrong (i.e., an SU detection insertion or deletion error), which will affect the features related to SU initial words. Recall also that the prosody model is built based on the features extracted around each word boundary; therefore, it does not account for the longer span prosodic features, which could be more useful for subtype detection. Since this is a new task, much research remains to be done to investigate what are the effective features and modeling approaches.

9.4 Edit Word Detection

9.4.1 Methods

For edit word and IP detection, an HMM as described in Chapter 5 is used as a baseline approach. In this section, two additional methods, the Maxent and CRF approaches, are examined for edit word and IP detection.

HMM for Edit Word Detection

For edit word detection, an HMM is first used for detecting IPs in edit disfluencies. The hidden event word-LM is trained from the joint word and edit IP sequence. There is no additional textual information (e.g., POS tags or automatically induced classes) used for the IP task. The prosody model is trained from a downsampled training set. Since the word-LM is generally undertrained and is able to only detect those repetitions that have occurred in the training set, a repetition detection model described in Section 5.2.3 is also used, which finds the repeated word sequences with possible filler words allowed after the edit IP. A rule-based approach is used to find the beginning of an edit disfluency after the IP is detected.

Maxent for Edit Word Detection

In the Maxent approach, first a Maxent classifier is used for a 3-way classification: SU, IP, or NULL. Then similarly to the HMM, heuristic rules are used to determine the onset of the reparandum. One advantage of this approach is that it jointly models SU and IP events. For example, if *"that is great. that is great"* has occurred in the training set, then the model will learn that these are two SUs, rather than an edit disfluency, even though the word sequence is repeated. In the repetition detection module in the HMM, we predefine some cue words that will be SUs and so would not be considered to be edit disfluencies (such as *'uhhuh uhhuh'*); whereas, the probabilistic Maxent model is able to learn these kinds of cue words from the training set and thus models them more elegantly. Also note that in the heuristic rules, the system SU hypotheses are used when determining the onset of a reparandum based on the IP hypotheses. For the Maxent approach, this SU information is generated directly by the Maxent classifier so the SU detection is tightly integrated in the system.

The features used in the Maxent model for the SU/IP/NULL detection task are as follows:

- All the features used for SU detection as described in Section 8.3.2.
- Repetition information. At each word boundary, this feature represents whether there is a repeated word sequence (up to 3 words) that ends at that point, with optional filler words allowed starting from that point.
- Fragment information. This feature represents whether a word is a fragment. Only in the reference transcription condition can this feature be triggered. In the speech recognition output condition, no word fragment information is provided.
- Filler words. This feature represents whether there is a pre-defined filler phrase after a word boundary.[1]
- Prosody posterior probabilities. A decision tree is trained for the binary classification task, IP or NULL, as in Section 5.2.3. The posterior probabilities are represented in a cumulative binning way.

CRF for Edit Word Detection

The CRF approach used for edit word detection finds the entire region of the reparandum, similarly to the named entity recognition task [148]. In this approach, each word has an associated tag, representing whether it is an edit word or not. The classes in the CRF edit word detection approach are: the beginning of an edit (B-E), inside of an edit (I-E), each of which has a possible IP associated with it (B-E+IP or I-E+IP), and outside of an edit (O). There is a total of 5 states in this model, as shown in Table 9.6. The following is an example of a transcription excerpt together with their class tags used in the CRF edit word detection model:

```
I       I    work    uh   i'm an analyst
B-E+IP I-E I-E+IP O    O    O   O
```

[1]This is not from the filler word detection results; rather a list of cue words is used.

```
and it   got    it got real rough
O   B-E  I-E+IP  O  O   O    O
```

The goal of this task is not only to find the reparandum extent, but also the IPs, including the internal IPs inside a complex edit disfluency. According to the annotation guideline and structural event detection task definition, IPs are annotated inside complex edit disfluencies, and they are also scored in the IP detection task. Therefore, IPs are included in the target class when using CRF for edit detection in order to identify the internal IPs inside complex edit disfluencies. For example, "*I I work*" in the above example is the reparandum in a complex edit disfluency, with an internal IP after the first "*I*".

Table 9.6
States and transitions used by the CRF for edit word and edit IP detection. The class tags are: the beginning of an edit (B-E), inside of an edit (I-E), each of which has a possible IP associated with it (B-E+IP or I-E+IP), and outside of an edit (O).

State number	Notation	Meaning	Possible state destinations
0	O	outside edit	O, B-E+IP, B-E
1	B-E+IP	begin edit with an IP	O, I-E+IP, I-E
2	B-E	begin edit	I-E+IP, I-E
3	I-E+IP	inside edit with an IP	O, B-E+IP, I-E+IP, I-E
4	I-E	inside edit	I-E+IP, I-E

The CRF model is able to learn valid state transitions from the training data. All of the possible states that a state can go to are shown in Table 9.6. Valid state transitions are guaranteed, i.e., only state 1 or 2 (the beginning of the edit) can transition to state 3 or 4 (inside a reparandum); whereas, state 0 cannot. An advantage of the CRF method is that it is a probabilistic model, and it provides a more principled way to represent this information than does using heuristic rules.

Features used in the CRF method are the N-grams of words and POS tags, turn change (as used in SU detection task), and all of the features used by the Maxent IP detection model that are not used for SU detection.

9.4.2 Edit Detection Results

The different models for edit word and IP detection are compared in Table 9.7 using the CTS data. Results are shown for both tasks using the NIST error rate. For the reference condition, the CRF is better at finding edit words, but poorer at IP detection compared to the HMM or Maxent methods. This ties into how the models are trained: the HMM and Maxent are trained to detect IPs, but the heuristic rules used may not find the correct onset for the reparandum; whereas, the CRF is trained to jointly detect the edit words and IPs and thus may not be as well trained for IP detection. However, on the STT condition, we observe that the CRF approach outperforms both the Maxent and HMM methods for both the edit word and edit IP tasks, suggesting that the CRF degrades less for the edit IP detection task on the STT condition. This is probably due to the fact that edit word and IP detection are mutually beneficial from the joint detection approach.

Table 9.7
Results (NIST error rate in %) for edit word and IP detection, using the HMM, Maxent, and CRF approaches on the reference and recognition output conditions of CTS data.

CTS				
Approaches	Edit word		Edit IP	
	REF	STT	REF	STT
HMM	54.33	85.32	33.21	73.66
Maxent	55.89	87.86	34.11	73.72
CRF	50.07	80.41	34.80	72.61

Table 9.8 shows the results for BN edit word detection on both the reference transcription and recognition output conditions, using the HMM and the Maxent approaches. A CRF was not used for the BN data due to the computational requirement of CRF. In addition, the edit disfluencies are very infrequent in BN. The Maxent approach yields better results for both edit word and IP detection than the HMM. Similar to previous findings, performance degrades severely in the STT condition compared to using reference transcriptions, with error rate increase more than that observed on the CTS edit word and IP detection tasks.

Table 9.8
Results (NIST error rate in %) for edit word and IP detection, using the HMM and Maxent approaches.

BN				
Approaches	Edit word		Edit IP	
	REF	STT	REF	STT
HMM	45.96	93.20	34.30	93.20
Maxent	43.00	89.86	30.89	87.54

9.5 Chapter Summary

We have described the investigation of new data, new tasks, and new approaches for structural event detection in the most recently RT-04 evaluation. Experiments on the new RT-04 data for SU boundary detection are consistent with previous findings. In addition, we applied the approaches used for SU detection (Maxent and CRF) to other tasks, such as SU subtype detection and edit word detection. Results have shown the Maxent and CRF outperform the prior HMM for edit word detection. The CRF approach for edit word detection avoids using ad-hoc rules as used in the HMM and Maxent approaches and allows more features to be easily incorporated. A two-step method for SU/SU-subtype detection yields a reasonable baseline sys-

tem performance, although additional research is needed to develop more effective features for this task, including textual and utterance-level prosodic features.

10. RELATED EFFORTS

In the previous chapters, we have described research related to each component of the structural event detection system, i.e., the prosody model, the language model, and their combination. In this chapter, we investigate factors that affect system performance, including the word error rate in recognition output and different methods to automatically derive speaker change information. In addition, we conduct a preliminary experiment to investigate whether acoustic and prosodic features can be used for word fragment detection. This is not a task currently defined in the EARS program. However, accurate identification of word fragments is helpful for edit word detection and can possibly improve speech recognition performance.

This chapter is organized as follows. In Section 10.1, we investigate some factors that impact the system performance for structural event detection. In Section 10.2, we describe preliminary experiments on word fragment detection. A summary of this chapter appears in Section 10.3.

10.1 Factors Impacting Performance

10.1.1 Word Error Rates (WER)

As we have already observed, there is a decreased accuracy on our structural event detection system when testing on the STT output compared to using human transcriptions, largely due to recognition errors. To understand just how much WER affects performance, we consider STT outputs from several recognition systems. Table 10.1 shows the SU and edit word detection results using several different STT systems on CTS and BN corpora. The WER for each STT system is indicated. For comparison, we also show results when using the reference transcription, which has

essentially a 0% WER. All of the structural event detection models are trained and tested on the RT-04 data set (see Table 9.1). The SU detection system is the majority vote of the HMM, Maxent, and CRF approaches for CTS, and the combination of the HMM and Maxent approaches via posterior probability interpolation. The edit word detection system is the CRF approach for CTS, and Maxent model for BN, as described in Section 9.4.

Table 10.1
SU and edit word detection results (NIST error rate in %) for CTS and BN, on REF and various STT conditions using the RT-04 data. For SU detection, results are reported for the SU boundary detection error. STT-1 and STT-2 are two different STT outputs, and the WER (%) for them is shown in the table.

Conditions		WER	SU boundary	Edit word
CTS	REF	0	26.21	50.07
	STT-1	14.9	39.18	80.41
	STT-2	18.6	44.26	80.92
BN	REF	0	47.15	43.00
	STT-1	11.7	59.73	89.86
	STT-2	15.0	62.67	91.40

As can be seen in Table 10.1, system performance degrades more when using a less accurate recognition output. Experimental results also show that word errors have a more negative impact on edit word detection than on SU detection. The relationship between WER and structural event detection performance appears to be non-linear, especially for edit word detection. For edit word detection, better STT accuracies only slightly improve performance, and there is a large gap between using the best STT output and the reference condition. This suggests that in the STT output more errors occur in the region of edit disfluencies and thus the word errors have a bigger impact on the edit word detection task. Also recall a main dif-

ference between the reference transcription and STT output for edit word detection is whether word fragment information is available or not. The lack of word fragment knowledge greatly impacts edit word detection on the STT condition. For SU detection, system performance is clearly impacted by different WERs. This makes sense since intuitively sentence initial and final words have a greater effect on the system performance for SU detection. Additionally, deletion errors in the STT output more are likely to occur in short SUs, such as backchannels, which have a more severe impact on SU detection than other deletion errors that occur in the middle of an utterance.

10.1.2 Speaker Label for SU Detection

As we have pointed earlier, speaker change is useful information for detecting SU boundaries. Speaker change affects the prosody model since it is in the prosodic feature set (as a single feature and one used to derive other features related to a "turn"). Looking at the feature usage by the decision trees for SU detection (Table 5.3 and Table 5.6), we notice that several features related to the speaker change are used by the trees. In the HMM, speaker change also affects the LM since it is used to chunk the word string into per-speaker based sequences: the continuous speech from the same speaker is concatenated into a sequence, to which a hidden event LM is applied. It is reasonable for a LM to hypothesize an SU at the end of a word sequence when there is a speaker change.

CTS and BN are processed differently to derive speaker turn change information. In CTS, since speech is recorded in separate channels, i.e., each channel corresponds to one speaker, the other channel needs to be considered to find whether there is a speaker change. Speech in each channel is segmented in places where there is a long pause, then the segments from the two channels are sorted using their beginning time. For one segment, if the following speech segment comes from the other channel, then a speaker change tag is recorded after this speech segment. Figure 10.1 shows

how the speaker change information is obtained for CTS. Segmentations 1, 4, and 5 are from speaker A; segmentations 2, 3, and 6 are from speaker B. After these segmentations are sorted based on their starting time, speaker change is added in those places marked with an arrow in the figure (i.e., after segment 1 for speaker A, and after segment 3 for speaker B). Note that there is a lot of overlapping in conversational speech (from the other channel), so this is simply an approximation of speaker change.

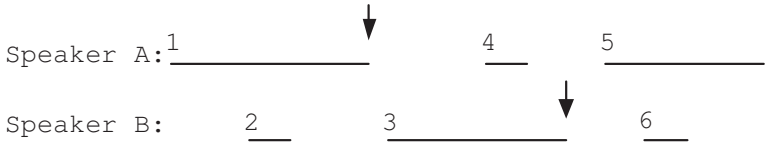

Fig. 10.1. An illustration of how speaker change is obtained for the CTS data. An arrow represents a speaker change after that segment.

In BN, there is only one channel and speaker information is unavailable; therefore, automatic speaker labeling is needed to identify speaker change. We investigated two different approaches to generate speaker labels for the pause-based segments described in Section 5.1.3. Note again that this investigation is conducted for the test set.

- Automatic speaker clustering: This method is used in speech recognition. An automatic clustering approach groups similar speech segments together for feature normalization or speaker adaptation [87]. The grouped segments have a cluster ID, which is used as a speaker label for the structural event detection tasks.

- Speaker diarization: An important task supported by the EARS MDE program is diarization. The goal is to add labels to the regions of the speech signal representing their sources, e.g., a particular speaker, music, or noise. We use the ICSI speaker diarization subsystem that generates the associated speaker labels for chunks of speech [151]. The algorithm first splits speech into k clusters

(generally greater than the expected number of speakers), then automatically clusters chunks of speech based on a metric similar to Bayes information metric, until there is no likelihood increase. The features used in the system are MFCC or PLP features depending on different broadcast shows. Speaker labels for the segments used in the event detection tasks can be obtained from this speaker diarization system output. Note that the pause-based segments do not align well with the chunks of speech corresponding to different speakers from the diarization system output. Each segment may contain multiple speakers based on the speaker diarization results. In this case, the speaker that has the majority of the speech in this segment is chosen as the speaker label for the event detection tasks.

Table 10.2 compares these two different methods to derive speaker labels for the BN SU detection task on the RT-04 test set. Results are reported for the reference condition using the improved HMM system (as shown in Table 9.2). We observe significant improvements when using the speaker labels derived from the speaker diarization output, suggesting that automatic speaker clustering is less appropriate for structural event detection. The goal of the automatic clustering is to cluster similar speakers together (based on acoustic similarity) for the purpose of recognizing words, and is less concerned with providing the correct speaker label.

Table 10.2
Comparisons of different ways to derive speaker labels on the RT-04 test set for the BN SU boundary detection task. Results are shown using the NIST error rate (%) for the HMM on the reference transcription condition.

BN	
Speaking Labeling Methods	SU Error Rate
Automatic clustering in STT	58.04
Diarization	51.76

So far we have only used the speaker diarization results to derive speaker labels for the pause-based segments of speech. We have conducted an experiment that uses the speaker diarization results to segment speech, rather than using a pause-based segmentation. However, this yields worse results than using the pause-based segments to which a speaker label is then assigned according to the speaker diarization results. Since the speaker diarization algorithm uses only acoustic information, it sometimes hypothesizes a speaker change in the middle of a continuous phrase from one speaker and thus can increase SU detection errors. A joint model for speaker recognition, speech recognition, and structural event detection is an important future direction.

10.2 Word Fragment Detection

In this section, we describe some preliminary experiments related to the detection of word fragments, which is not evaluated as a structural event detection task by itself in the EARS program. A word fragment, also called a partial word, occurs when a speaker cuts off speaking in the middle of a word. Word fragments indicate the presence of disfluencies; however, most current speech recognizers do not detect them, thus important information is lost for disfluency detection. Accurate word fragment detection should also be very important for speech recognition.

10.2.1 Introduction

Word fragments occur frequently in spontaneous speech, and are good indicators for speech disfluencies [13, 56]. Levelt found the percentage of the disfluencies that contain a word fragment to be 22% for a pattern description task in Dutch [67]; Lickley reported 36% for casual conversations in British English [152]; Bear et al. found 60% for the ATIS corpus [49]. We examined 83 conversations of Switchboard data and found that about 17% of the disfluencies contain word fragments. However, accurate identification of word fragments in a speech recognizer is still an unsolved

problem. In most cases, they are simply treated as out-of-vocabulary words or are often incorrectly recognized as words in the vocabulary. This not only affects accurate recognition of neighboring words, but also fails to provide the important information that a word fragment is present, which is important for detecting an interruption point in a disfluency.

The following is an example of the human transcription and the speech recognition output from the Switchboard corpus:

Human transcription:

and it's all just you know i've just eating more sort of eat to my apper- appetite

Recognizer output:

and it's all just see now i'm just eating more sort of need to my out bird's appetite

We can see that in the recognition output, the word fragment *"apper-"* is incorrectly recognized as two words in the vocabulary. Additionally, due to the failure to identify the word fragment *"apper-"*, it will be extremely difficult to identify the disfluency in the recognition results.

The study of word fragments has been conducted across several disciplines. Psychologists and linguists [38] suggest that speakers rarely interrupt a word when it is correct on its own, but they often do so when it is not. When a word is complete, the speakers are committing themselves to its correctness (at least at that moment).

While linguists and psycholinguists have considered this problem from the production point of view, computational linguistics have investigated this problem with the goal of better speech recognition and disfluency detection. As noted by Beat et al. in [49], knowledge about the location of word fragments would be an invaluable cue to both detection and correction of disfluencies. Heeman and Allen [56] proposed an integrated model for the detection of speech repairs that incorporates word fragments as an important feature. Nakatani and Hirschberg [13] proposed a "speech-first" model for the detection of speech repairs using acoustic-prosodic cues. They found that the presence of word fragments is an important indicator of speech

repairs, along with the other prosodic-acoustic features such as silence duration, energy, and pitch. They analyzed the properties of word fragments, for example, the distribution of the fragments in syllable length, the distribution of initial phonemes in the fragments, and some acoustic cues (glottalization and coarticulation) in the fragments. Although the role of word fragments as an indicator of disfluencies is emphasized, they did not address the problem of how to detect the occurrence of word fragments, but only suggest that a word-based language model for word fragment detection is unlikely to be effective. O'Shaughnessy [65] observed in the ATIS corpus that when speaker stops in the middle of a word and then resumes speaking with no changed or inserted words (i.e., a repetition), the pause lasts 100-400 ms in 85% of the examples (with most of the remaining examples having pause of about 1 second in duration). He also found that three-fourths of the interrupted words do not have a completion of the vowel in the intended word's first syllable (i.e., the speaker stops after uttering the first consonant).

Although word fragments should play an important role for disfluency processing in spontaneous speech, the identification of word fragments is yet to be tackled by the speech community. It is infeasible to treat word fragments as regular words by including all the partial words in the dictionary. Furthermore, it may be quite difficult to train a good model to cover all the word fragments due to the variability of the possible partial words. Rose and Riccardi modeled word fragments by using a single word fragment symbol *frag* in the system "How May I Help You" [153]. Their system was improved by explicitly modeling word fragments along with the filled pauses and non-speech events; however, they did not directly report on the impact of modeling word fragments.

We investigate the problem of word fragment detection by using speech analysis. Our goal in this study is to identify reliable acoustic-prosodic features for word fragment detection.

10.2.2 Acoustic and Prosodic Features

Our hypothesis is that there are indicative prosodic cues and voice quality characteristics at the boundary of word fragments; hence, our approach is to extract a variety of acoustic and prosodic features and build a classifier using these features for the automatic identification of word fragments. The same prosodic features as used for the structural event detection in Chapter 5 are used. In addition, we believe that when the speaker suddenly stops mid-word, the voice quality is more likely to change, so a new set of voice quality related features was investigated for word fragment detection.

Human speech sounds are commonly considered to result from a combination of a sound energy source modulated by a transfer (filter) function determined by the shape of the vocal tract. As the vocal cords open and close, puffs of air flow through glottal opening. The frequency of these pulses determines the fundamental frequency of the laryngeal source and contributes to the perceived pitch of the produced sound.

The voice source is an important factor affecting the voice quality, and thus our investigation focuses on the voice source characteristics. The analysis of voice source has been done by inverse filtering the speech waveform, analyzing the spectrum, or by directly measuring the airflow at the mouth for non-pathological speech. A widely used model for voice source is the Liljencrants-Fant (LF) model [154, 155]. Research has shown that the intensity of the produced acoustic waveform depends more on the derivative of the glottal flow signal than the amplitude of the flow itself.

An important representation of the glottal flow is given by the Open Quotient (OQ). OQ is defined as the ratio of the time in which the vocal folds are open to the total length of the glottal cycle. From the spectral domain, it can be empirically formulated as [156]:

$$5.5 \times OQ = log((H_1^* - H_2^* + 6)/0.27) \tag{10.1}$$

where H_1^* and H_2^* are the amplitudes of the first and the second harmonics of the spectrum.

Different phonation types, namely, modal voicing, creaky voicing, and breathy voicing, differ in the amount of time that the vocal folds are open during each glottal cycle. In modal voicing, the vocal folds are closed during half of each glottal cycle. In creaky voicing, the vocal folds are held together loosely resulting in a short open quotient. In breathy voicing, the vocal folds vibrate without much contact thus the glottis is open for a relatively long portion of each glottal cycle. We think it is possible that there are some creaking or breathy voicing when a word fragment occurs.

For the word fragment detection task, the following voice quality related features are investigated.

- **Jitter** is a measure of perturbation in the pitch period that has been used by speech pathologists to identify pathological speech [157]; a value of 0.01 represents a jitter of one percent, a lower bound for abnormal speech.

  The value of jitter is obtained from the speech analysis tool *Praat* [158]. The pitch analysis of a sound is converted to a point process, which represents a sequence of time points, in this case the times associated with the pitch pulses. The periodic jitter value is defined as the relative mean absolute third-order difference of the point process (or the second-order difference of the interval process).

  $$jitter = \frac{\sum_{i=2}^{N-1} |2 \times T_i - T_{i-1} - T_{i+1}|}{\sum_{i=2}^{N-1} T_i} \tag{10.2}$$

  where T_i is the ith interval and N is the number of the intervals of the point process. If no sequence of three intervals can be found whose durations are between the shortest period and the longest period, the result is undefined [158].

- **Spectral tilt** is the overall slope of the spectrum of a speech or instrument signal. For speech, it is responsible for the prosodic features of accent, in that a speaker modifies the tilt (raising the slope) of the spectrum of a vowel to put stress on a syllable. In breathy voice, the amplitude of the harmonics in the spectrum drops off more quickly as the frequency increases than in the modal

or creaky spectra, i.e., breathy voice has a greater slope than creaky voice. Spectral tilt is measured in decibels per octave. A linear approximation of the spectral envelope is used to measure spectral tilt.

- **OQ** is defined in Equation (10.1). It is derived from the difference of the amplitude of the first and the second harmonics of the spectral envelope of the speech data. Studies have shown that the difference between these two harmonics (and thus the OQ) is a reliable way to measure the relative breathiness or creakiness of phonation [159]. Breathy voice has a larger OQ than creaky voice. As an approximation, $F0$ and $2 \times F0$ are used for the first and the second harmonics in the spectrum.

10.2.3 Experiments

Experimental Setup

Our goal is to identify reliable acoustic-prosodic features for word fragments. Similar to the event detection tasks in Chapter 4, the task of word fragment identification is viewed as a statistical classification problem, i.e., for each word boundary, a classifier determines whether the word before the boundary is a word fragment or not using acoustic-prosodic features.

Part of the Switchboard corpus is used for our experiments.[1] In the human transcriptions, word fragments are identified (around 0.7% of the words are word fragments). 80% of the data is used as the training data, with 20% remaining for testing. At each boundary location, prosodic features and voice quality measures are extracted as described in the previous section. A decision tree classifier was trained from a downsampled training set that contains 1438 samples, and tested on a downsampled test set with 288 samples (50% of the samples in the training and test set are word fragments).

[1]See http://www.icsi.berkeley.edu/~yangl/thesis/fragment.html for information about the conversations used for this study.

Experimental Results

Table 10.3 shows the confusion matrix results for the classification task of word fragment versus complete words using the downsampled data. The precision and recall for this fragment detection task are 74.3% and 70.1% respectively. The overall accuracy for all the test samples is 72.9%, which is significantly better than the chance performance of 50%. These results suggest that acoustic-prosodic features are effective for word fragment detection.

Table 10.3
Word fragment detection results (in confusion matrix) on the downsampled data of Switchboard corpus.

		hypothesis	
		complete	fragment
reference	complete	109	35
	fragment	43	101

An inspection of the decision tree's feature usage in the results reveals the most effective features for distinguishing word fragments from complete words. Table 10.4 reports the feature usage for this word fragment detection task. Figure 10.2 shows the pruned decision tree for this task. Among the voice quality features, jitter is queried the most by the decision tree. This is likely due to the fact that when a speaker suddenly cuts off in the middle of the word, there is abnormality of the vocal fold (e.g., the pitch period) that is captured by jitter. The average OQ (AVG_OQ) is also chosen as a useful feature, suggesting that a mid-word interruption generates some creaky or breathy voice. The questions produced by the decision tree show that word fragments are hypothesized if the answer is positive to the questions such as 'jitter $>$ 0.018053?', 'average OQ $<$ 0.020956?', or 'average OQ $>$ 0.60821?'. Speech with these attributes has an abnormal voice quality. We have also conducted

classification experiments using only two features, jitter and average OQ, and a classification accuracy of 68.06% is obtained.

Table 10.4
Feature usage (%) for word fragment detection using the Switchboard data.

Feature	**Usage (%)**
JITTER	27.2
ENERGY_PATTERN_BOUNDARY	24.1
F0K_WIN_DIFF_LOHI_N	23.8
AVG_OQ	14.7
TURN_CNT	8.4
PAU_DUR	1.8

From Table 10.4, we also observe that one energy feature and one F0 feature are queried frequently. However, we may need to be careful of interpreting these prosodic features, because some word fragments are more likely to have a missing (or undefined) value for the stylized F0 or energy features (due to their short duration and unvoiced frames). For example, in one leaf of the decision tree, a word fragment is hypothesized if the energy slope before the boundary is an undefined value (as shown in Figure 10.2, the question is 'ENERGY_PATTERN_BOUNDARY in Xr, Xf?', where 'X' means undefined value).

Notice that the usage of the pause feature is very low, even though a pause is expected after a sudden closure by the speaker. One reason for this is that the recognizer is more likely not to generate a pause in the phonetic alignments when the pause after the mid-word interruption is very short. For example, around two thirds of the word fragments in our training and test set are not followed by a pause based on the alignments. Additionally, there are many other places (e.g., sentence boundaries or filled pauses) that are often followed by a pause. Therefore, being

```
JITTER < 0.049782:  0
|  ENERGY_PATTERN_BOUNDARY in rf,fr,fX,rr,ff,rX :  0
|  |  F0K_WIN_DIFF_LOHI_N < -0.093224:  0
|  |  |  AVG_OQ < 0.60821:  0
|  |  |  |  TURN_CNT < 13.5:  FRAGMENT
|  |  |  |  |  ENERGY_PATTERN_BOUNDARY in rf,fX,Xr,ff,Xf :  0
|  |  |  |  |  |  JITTER < 0.018053:  0
|  |  |  |  |  |  JITTER >= 0.018053:  FRAGMENT
|  |  |  |  | ENERGY_PATTERN_BOUNDARY in fr,rr,rX: FRAGMENT
|  |  |  |  TURN_CNT >= 13.5:  0
|  |  |  |  |  AVG_OQ < 0.20956:  FRAGMENT
|  |  |  |  |  AVG_OQ >= 0.20956:  0
|  |  |  AVG_OQ >= 0.60821:  FRAGMENT
|  |  F0K_WIN_DIFF_LOHI_N >= -0.093224:  0
|  ENERGY_PATTERN_BOUNDARY in Xr,Xf :  FRAGMENT
JITTER >= 0.049782:  FRAGMENT
|  F0K_WIN_DIFF_LOHI_N < -0.14995:  FRAGMENT
|  F0K_WIN_DIFF_LOHI_N >= -0.14995:  FRAGMENT
|  |  ENERGY_PATTERN_BOUNDARY in rf,fX,rr,ff,rX :  0
|  |  |  PAU_DUR < 12.5:  0
|  |  |  PAU_DUR >= 12.5:  FRAGMENT
|  |  ENERGY_PATTERN_BOUNDARY in fr,Xr,Xf :  FRAGMENT
```

Fig. 10.2. The pruned decision tree used to detect word fragments. The decision is made in the leaf nodes; however, in the figure the decision for an internal node in the tree is also shown.

followed by a pause cannot always accurately distinguish between a word fragment and other complete words.

10.3 Chapter Summary

We have investigated a few factors that impact the structural event detection system performance, including the WER of the speech recognition results and the speaker labeling approaches. Better recognition output has a greater impact on SU detection than edit word detection, suggesting word errors may occur more often in the edit disfluency region. The difference across BN and CTS also suggests that edit word detection is relatively easier for BN human transcription condition than CTS

because of different styles of the two corpora. Comparison of two different speaking labeling methods for BN SU detection task has shown that using the speaker labels that are derived from the speaker diarization system is more appropriate than using the automatic speaker clustering as used in speech recognition.

Accurate identification of word fragments would be very helpful for a disfluency detection algorithm because the occurrence of word fragments is a good indicator of speech disfluencies. We investigated the problem of word fragment detection using acoustic and prosodic features. Preliminary experimental results show that some acoustic-prosodic features provide useful information for word fragment detection. As a first approximation of the characterization of word fragments via the acoustic-prosodic cues, we find these results encouraging. They offer an alternative approach to build acoustic models and suggest speech analysis can be quite relevant to building speech recognition systems that are more capable of recognizing fragments.

The experiments for word fragment detection are very preliminary. Investigations using large corpora and more sophisticated versions of our measurements, especially for the voice quality measurements, should be made. Additionally, experiments were conducted using only a downsampled data set due to the highly skewed data distribution. The current word fragment detection method would generate many false alarms in the real test situation, i.e., on non-downsampled data. We should also investigate the performance of our algorithm when applying it directly to speech recognition results.

11. FINAL REMARKS

11.1 Impact on Other Research Efforts

The research in this thesis has had impact on related research efforts; hence, in this section, we discuss three research efforts that have benefited from this work. One uses the structural event detection system output to further improve speech recognition by modifying speech segments. The other two involve applying the techniques that have been developed in this thesis to other corpora: a multimodal corpus and a multiparty meeting corpus.

11.1.1 Using Structural Event Information for Word Recognition

Theoretically a joint model is needed to recognize words and structural events simultaneously. However, due to the lack of a general framework for this purpose, we have utilized a two-step approach for investigating the impact of structural event information on speech recognition.

We believe that linguistic segments should be better than acoustic segments for speech recognition. For example, in LM rescoring, intuitively the initial word of a sentence should not be dependent on the final words of the previous sentence. Hence, information about sentence boundaries should be particularly beneficial to a LM. Yet currently the LM treats each acoustic segment used in speech recognition as a 'sentence'. The error analysis in [160] showed that the errors near the sentence boundaries are higher than in other places. This motivates our investigation of

whether SU boundary information can be used for generating better segments to improve recognition accuracy.[1]

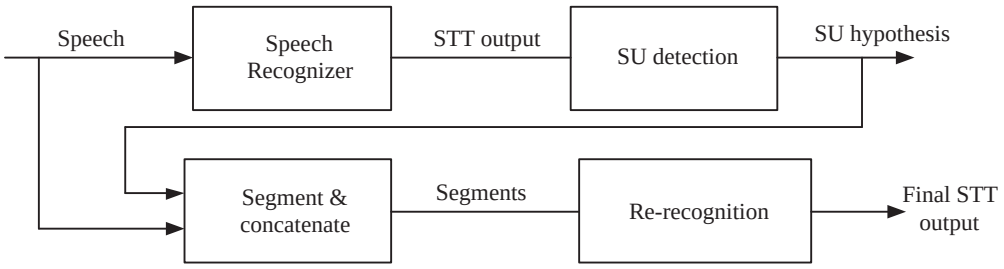

Fig. 11.1. Using SU information for re-recognition in BN.

Figure 11.1 shows how SU information is fed back to a recognizer in a two-step approach. First the SU detection system is applied to the recognition output. Then speech is resegmented using the hypothesized SUs and the word alignments. See [161] for details on the segmentation step. Finally the new segments are used for another pass of speech recognition.

Evaluations were conducted using the RT-03 development set of BN data.[2] Table 11.1 shows the WER after re-segmenting the speech using the SU detection results and re-recognition. Results are shown using both reference SUs and the system generated SUs. The SU detection system is the improved HMM system. Also shown in the table is the baseline recognition results using acoustic segments obtained from a speech/non-speech detector. The recognizer used in these experiments is a simplified version of the SRI BN recognizer [162]. As observed from the table, using SU information to chop the speech yields better segments and thus better recognition performance. Also using the reference SUs results in better recognition accuracies than using automatically detected SUs (due to the SU insertion and deletion errors in the latter). These results suggest that linguistic segments provide an important alternative to acoustic segments for recognition. It also highlights the importance

[1]This is the work of Sebastien Coquoz at ICSI [161]. This result is included here to show the impact of our system.

[2]This is half of the RT-03 test set discussed in Table 3.2.

of the interaction between structural event detection and speech recognition. This result is quite preliminary because of the small testing set used, and thus additional investigation is needed. Additionally, this is only a loosely coupled approach, i.e., run recognition first, then detect structural events based on the recognition results, and finally re-run recognition using the structural event hypotheses. A more tightly coupled approach should result in a better performance.

Table 11.1
WER (%) when SU information is fed back to re-segment and re-recognize speech, compared to the baseline using the acoustic segments, evaluating on half of the RT-03 BN data.

	WER
Use reference SUs	13.0
Use automatically generated SUs	13.3
Baseline: acoustic segments	14.0

11.1.2 SU Detection in a Multi-modal Corpus

Generalization of CTS Models

Our SU detection model was applied to the wombat data set, which was collected to investigate multimodality in dialog (see http://vislab.cs.wright.edu/KDI/). The audio was digitally recorded using a unidirectional boom-mounted microphone placed at a fixed distance for each interlocutor in a somewhat noisy laboratory environment. The dialog concerns development of a plan to catch a family of intelligent wombats that has taken over the theater in the town of Arlee in order to send them back to Australia. Transcripts were force aligned to the audio signal and hand corrected. The speech was annotated using the LDC V5 annotation guideline as for CTS and BN corpora [71]. The task domain differs from CTS and BN in that it involves the

development of a plan, and the audio is far noisier. There are three dialogs with six distinct participants. Each recording is about 10 minutes of speech.

The same models as used for the RT-03 CTS data are used, since there is no similar multimodal corpus available for modeling training. Therefore, there is some domain mismatch between the training and the test conditions. The reason we chose to use CTS is because both corpora involve conversational speech. This is reflected by the percentage of the SUs in the wombat data and the CTS data, roughly 14% interword boundaries are SU boundaries.

Table 11.2 shows the SU boundary detection results for this task using the HMM and the Maxent approaches alone, and in combination. Note again that this is conducted using the reference transcription. We observe similar patterns as on the CTS reference transcription condition (Table 8.2); however, the error is much greater. The noisier recording conditions and the new task domain challenges both the textual and prosodic knowledge sources.

Table 11.2
SU detection results (NIST error rate in %) on the Wombat data. * Note that the combined result is not shown when using textual information only, in order to make results in parallel to the results in Chapter 8 (Table 8.2 and Table 8.4).

	HMM	Maxent	Combination
Textual+prosodic	45.25	44.70	43.49
Textual only	63.36	58.72	*

Combining Speech Features with Gesture Features

In this thesis we have only used the recorded speech data for structural event detection. However, humans use every mode they can, such as gesture and eye gaze, to convey information and better understand each other.

The HMM was applied to the multimodal wombat corpus, using prosodic information, gestural information, and lexical cues for finding SUs.[3] Experiments have confirmed that each knowledge source provides additional information, and their combination achieves the best performance [163]. Similarly to the prosody model, the sampling and bagging techniques have been used for building the gestural model and have proven effective. Preliminary experiments have also shown that prosodic features and gestural features do not combine as well when the features are jointly modeled by the decision trees as when they are loosely combined by interpolating the posterior probabilities of the decision trees from each source. This work suggests that the modeling approach we have developed for SU detection in speech is effectively extended to other knowledge sources.

11.1.3 Dialog Act Detection in Meeting Corpus

There has been a growing interest in automatic processing of multiparty meetings. Common goals in addition to word recognition include automatic browsing, retrieval, question-answering, and summarization. Such tasks would require segmenting continuous speech into functional units, or dialog acts (DAs). These DAs are similar in a sense to the SUs investigated in conversational speech.

We explore both DA boundary detection and its subtype detection using the ICSI Meeting Corpus [164], which includes 75 naturally occurring meetings containing roughly 72 hours of multitalker speech data and associated human-generated transcriptions.[4] The corpus was hand-annotated for dialog acts as described in detail in [166, 167]. We grouped various DA labels into five broad categories: statements, questions, backchannels, fillers, and disruptions. The meeting data was recognized by an SRI recognizer [162], which was trained on CTS data and yielded a WER of about 39% on the entire meeting corpus. The corpus is split into 51 meetings for training, 11 for development, and the remaining 11 meetings for testing.

[3]This is joint work with Lei Chen at Purdue University [163].
[4]This is joint work with Jeremy Ang at ICSI [165].

An HMM as described in Chapter 4 is used for DA boundary detection, similarly to SU boundary detection. For this experiment, only a single pause feature is used as the prosodic feature, and a single decision tree is trained as the prosody model. A hidden event LM is trained to model the joint word and DA event sequence. Table 11.3 shows the DA boundary detection results using the human transcription or STT output. About 16.2% of the word boundaries are at the end of a DA, which is comparable to the percentage of SUs in CTS. As can be seen, the prosody model contributes more on the STT condition to the combined results than the reference condition, as was found in the CTS SU detection results.

Table 11.3
DA boundary detection results (NIST error rate in %) on ICSI Meeting data. Results are for the reference transcriptions (REF) and STT output, using the pause decision tree (pause DT) model, hidden event LM, and the HMM combination of them.

	pause DT	LM	HMM Combination
REF	56.67	45.92	43.54
STT	58.80	61.81	54.72

A Maxent classifier like that used for SU subtype detection is built for DA subtype detection. The same lexical features as for the SU subtype detection task are used. However, for this experiment the prosodic features (duration, pitch, and energy) are extracted from the whole DA unit, unlike those used in SU subtype detection task, which focuses only on the features associated with each word boundary. Table 11.4 shows the DA subtype classification accuracy results using the reference DA boundaries for both the human transcriptions and recognition output.[5] Accuracy is measured as the percentage of the DAs that are labeled with the correct class. Results are shown when using only word-based features, and the combined

[5]The reference DA boundaries for the recognition output are generated by aligning the recognition words with reference transcriptions plus the DA annotations.

word-based features and the binned posterior probabilities from the decision tree. As can be seen, classification accuracy is significantly better than chance performance, and incorporating the prosodic information improves classification accuracy. See [165] for more details when automatic DA boundary detection is used, which uses a similar two-step approach as used for the SU/SU-subtype detection as described in Chapter 9.

Table 11.4
DA subtype classification accuracy (%) using the reference DA boundaries of the ICSI Meeting corpus using the human transcriptions and recognition output. Two conditions are used: word-based features only, and the combined word-based features and the binned posterior probabilities from the decision tree (DT). Chance performance is obtained when the majority type (statement) is hypothesized for each DA.

	Chance	Word features	Word + DT posteriors
REF	55.08	79.53	81.18
STT	57.07	72.33	73.96

11.2 Summary of Experiments

A systematic study has been conducted on automatic detection of structural events in speech, namely, SU, edit disfluency, and filler detection, with SU and disfluencies being of the main focus. Experiments were conducted on two corpora (conversational speech and broadcast news speech) and two types of transcriptions (human transcriptions and speech recognition output).

Experiments show that speakers use prosodic cues to resolve ambiguities in speech, signal the end of an SU, identify the interruption point in an utterance, or mark discourse structure. Prosodic features provide a valuable knowledge source for automatic structural event detection, with different prosodic features found to

be effective for different corpora and speaking styles. Additionally, performance of the prosody model varies for different structural event detection tasks because of the different event distributions and the inherent characteristics of the tasks.

Textual information was found to be an important knowledge source for detecting structural events. The hidden event LM has performed reasonably well for such tasks. The word-based N-gram has been extended, to include more textual cues, such as POS, automatically induced classes, and syntactic chunks. Some lexical features remain to be investigated, such as parsing information. A repetition pattern detector was also developed for edit disfluency detection. Generally when used alone, a word-based LM is superior to the prosody model for the structural event detection tasks. However, the combination of the prosody model and LM usually outperforms either individual model, suggesting the importance of integrating multiple knowledge sources for improving system performance. Both the prosody model and LMs degrade when using the recognition output due to the word errors and incorrect phone alignments. We have found that the LM has greater relative error increase when using recognition output instead of human transcriptions than the prosody model. Several factors that have an impact on the event detection performance have been studied, including the recognition error rate and different speaker labeling methods.

We have investigated the imbalanced data set problem encountered in training the prosody model. A variety of sampling and bagging methods were evaluated for the SU boundary detection task. If classification accuracy is the performance measure, then using the original training set yields the best results when the prosody model is used alone; however, if the performance metric is ROC or if the minority class is deemed of more interest, then sampling methods are more important. Bagging generates multiple classifiers, reduces the variance, and significantly improves the system performance. Studies across the SU and IP tasks have also highlighted their inherent differences with respect to machine learning methods.

Three modeling approaches have been compared for combining knowledge sources, the HMM, Maxent, and CRF approaches for SU detection. An HMM is a generative

approach; whereas, the Maxent and CRF are discriminative models that directly estimate the posterior probabilities of events given observations (textual and prosodic features). The Maxent approach only models local information; whereas, both the HMM and CRF are able to model the entire sequence. The Maxent and CRF are better at integrating overlapping textual information, but currently only use the binned posterior probabilities from the prosody model and thus ignore fine-grained knowledge about prosody, one possible reason that their performance degrades more in the STT conditions. The HMM makes more effective use of the prosody model, but does not jointly model various textual features. A model combining all these approaches generally achieves the best performance.

These approaches were also examined for edit disfluency detection. Both the HMM and the Maxent first detect the interruption points and then apply heuristic rules for determining the onset of the reparandum. The CRF provides a more principled way to incorporate knowledge sources in a probabilistic way for detecting the extent of reparandum, avoiding the ad-hoc rules used in the HMM and Maxent methods.

11.3 Contributions

The contributions of this thesis are three fold.

- A systematic and comprehensive investigation of structural event detection in speech has been conducted. This includes finding more indicative prosodic features, effectively using more textual information, and developing better methods for constructing each model and combining different models than in prior work. Our investigations across different corpora, using both human transcriptions and recognition output, and three types of structural events have enabled us to create better models for structural events.

- This thesis also emphasizes the importance of using knowledge from other disciplines to improve spontaneous speech event processing. First, machine

learning techniques are crucial for training improved prosody models from an imbalanced data set and for combining multiple classifiers. Second, methods from natural language processing are important for more effectively exploiting textual information. Finally, speech analysis measurements offer additional valuable features for extending our prosodic feature set. The knowledge from these disciplines has been beneficial to our ultimate goal of better modeling speech events.

- The research done in this thesis has proven helpful to several related research efforts. A preliminary study has shown that using the structural event output can improve speech recognition accuracy. Ongoing research attempts to tightly integrate the event information into a statistical LM for speech recognition. The methods developed in this thesis have been successfully applied to two other corpora: prosodic and gestural features are combined with the hidden event LM for SU detection in a multimodal corpus; the SU/SU-subtype detection approach has been successfully utilized for dialog act detection and in a multiparty meeting corpus.

11.4 Future Work

There are many important future directions for this work on structural event detection in speech. A few directions that are most relevant to the research in this thesis are listed below.

Additional acoustic-prosodic features should be evaluated for their effectiveness in detecting different structural events. Long-span features that capture suprasegmental information are an important direction. Speaker dependent modeling is an interesting and important avenue for improving our models, since different speakers have different speaking styles. For example, some speakers do not pause between SUs, while others use very specific discourse markers. A word-dependent prosody model may also be helpful for discourse marker detection.

Another direction would be to develop a better modeling approach for the prosody model. We have only extensively investigated the imbalanced date set problem when training the decision tree prosody model. Other directions include using other machine learning techniques, such as support vector machines (which have shown a great success in many applications) to implement the classifier, or methods that more effectively combine multiple classifiers.

For textual information, a future direction is to include more syntactic structure in the knowledge sources. Simple N-gram LMs (even those that are POS- or chunk-based) cannot capture such higher level information. In our error analysis, we have found that there are many insertion errors at the phrase boundaries. Hence, parse structure information would be helpful for eliminating these insertion errors. In addition, the Maxent and CRF approaches have proven to be more effective at modeling correlated textual features, yet they currently use only binned prosody model posterior probabilities. Methods for direct incorporation of prosodic features into the Maxent and CRF approaches need to be investigated.

Since speech recognition errors have a significant impact on the system performance, examining the use of confusion networks or word lattices [168] can leverage multiple recognizer hypotheses and thus may improve performance for both structural event detection and speech recognition. A tightly coupled framework for both the structural event detection and speech recognition is a final important future direction for the detection of the events themselves.

Most of the future work involves creating better models. We believe that although the data-driven approach used in this thesis is important for building automatic structural event detection systems, measurement studies are an important avenue for investigating what features are most important for detecting these events.

Structural event detection is an important bridge that links speech recognition and downstream language processing modules. Therefore investigating the impact of structural event detection on downstream applications, such as parsing, machine

translation, summarization, and information extraction, is an important future direction.

LIST OF REFERENCES

LIST OF REFERENCES

[1] D. Jones, F. Wolf, E. Gibson, E. Williams, E. Fedorenko, D. Reynolds, and M. Zissman. Measuring the readability of automatic speech-to-text transcripts. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 1585–1588, 2003.

[2] M. Gregory, M. Johnson, and E. Charniak. Sentence-internal prosody does not help parsing the way punctuation does. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, 2004.

[3] J. G. Kahn, M. Ostendorf, and C. Chelba. Parsing conversational speech using enhanced segmentation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, 2004.

[4] W. N. Campbell. Durational cues to prominence and grouping. In *Proceedings of ECSA Workshop on Prosody*, pages 38–41, Lund, Sweden, 1993.

[5] R. Lickley and E. Bard. On not recognizing disfluencies in dialog. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1876–1879, 1996.

[6] J. R. De Pijper and A. A. Sanderman. On the perceptual strength of prosodic boundaries and its relation to suprasegmental cues. *Journal of the Acoustical Society of America*, 96(4):2037–2047, October 1994.

[7] D. Hirst. Peak, boundary and cohesion characteristics of prosodic grouping. In *Proceedings of ECSA Workshop on Prosody*, pages 32–37, Lund, Sweden, 1993.

[8] P. J. Price, M. Ostendorf, S. Shattuck-Hufnagel, and C. Fong. The use of prosody in syntactic disambiguation. *Journal of the Acoustical Society of America*, 90(6):2956–2970, 1991.

[9] S. Potisuk. *Prosodic Disambiguation in Automatic Speech Understanding of Thai*. PhD thesis, Purdue University, 1995.

[10] D. R. Scott. Duration as a cue to the perception of a phrase boundary. *Journal of the Acoustical Society of America*, 71(4):996–1007, 1982.

[11] M. Swerts. Prosodic features at discourse boundaries of different strength. *Journal of the Acoustical Society of America*, 101(1):514–521, January 1997.

[12] E. Shriberg, A. Stolcke, D. Hakkani-Tur, and G. Tur. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, pages 127–154, 2000.

[13] C. Nakatani and J. Hirschberg. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustical Society of America*, pages 1603–1616, 1994.

[14] R. Kompe. *Prosody in Speech Understanding System*. Springer-Verlag, 1996.

[15] Y. Gotoh and S. Renals. Sentence boundary detection in broadcast speech transcripts. In *Proceedings of ISCA Workshop: Automatic Speech Recognition: Challenges for the new Millennium ASR-2000*, pages 228–235, 2000.

[16] J. Kim and P. C. Woodland. The use of prosody in a combined system for punctuation generation and speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2757–2760, 2001.

[17] H. Christensen, Y. Gotoh, and S. Renal. Punctuation annotation using statistical prosody models. In *ISCA Workshop on Prosody in Speech Recognition and Understanding*, 2001.

[18] E. Shriberg and A. Stolcke. A prosody-only decision-tree model for disfluency detection. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2383–2386, 1997.

[19] M. Meteer and R. Iyer. Modeling conversational speech for speech recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996.

[20] D. D. Palmer and M. A. Hearst. Adaptive sentence boundary disambiguation. In *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing*, pages 78–83, 1994.

[21] J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington D.C.*, pages 16–19, 1997.

[22] H. Schmid. Unsupervised learning of period disambiguation for tokenization. University of Stuttgart, Internal Report, 2000.

[23] D. J. Walker, D. E. Clements, M. Darwin, and J. W. Amtrup. Sentence boundary detection: A comparison of paradigms for improving MT quality. In *Proceedings of MT Summit VIII: Santiago de Compostela*, 2001.

[24] D. Beeferman, A. Berger, and J. Lafferty. Cyperpunc: A lightweight punctuation annotation system for speech. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 1998.

[25] M. Stevenson and R. Gaizauskas. Experiments on sentence boundary detection. In *Proceedings of the North American Chapter of the Association for Computational Linguistics annual meeting*, pages 24–30, 2000.

[26] M. Fach. A comparison between syntactic and prosodic phrasing. In *Proceedings of the European Conference on Speech Communication and Technology*, 1999.

[27] S. Abney. Chunks and dependencies: Bring processing evidence to bear on syntax. *Computational Linguistics and the Foundations of Linguistic Theory*, pages 145–164, 1995.

[28] K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirsberg. ToBI: A standard for labeling english prosody. In *Proceedings of the International Conference on Spoken Language Processing*, pages 867–870, 1992.

[29] C. J. Chen. Speech recognition with automatic punctuation. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 447–450, 1999.

[30] A. Stolcke, E. Shriberg, R. Bates, M. Ostendorf, D. Hakkani, M. Plauche, G. Tur, and Y. Lu. Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2247–2250, 1998.

[31] E. Shriberg and A. Stolcke. Prosody modeling for automatic speech recognition and understanding. In *Proceedings of the Workshop on Mathematical Foundations of Natural Language Modeling*, 2002.

[32] J. Ang, R. Dhilon, A. Krupski, E. Shriberg, and A. Stolcke. Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2037–2040, 2002.

[33] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26:339–373, 2000.

[34] J. Huang and G. Zweig. Maximum entropy model for punctuation annotation from speech. In *Proceedings of the International Conference on Spoken Language Processing*, pages 917–920, 2002.

[35] National Institute of Standards and Technology. RT-03F workshop agenda and presentations. http://www.nist.gov/speech/tests/rt/rt2003/fall/presentations/, November 2003.

[36] D. Wang and S. S. Narayanan. A multi-pass linear fold algorithm for sentence boundary detection using prosodic cues. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2004.

[37] G. S. Dell. A spreading activation theory of retrieval in sentence production. *psychological Review*, pages 283–321, 1986.

[38] W. Levelt. *Speaking: From Intention to Articulation*. Cambridge, MA: MIT press, 1989.

[39] D. G. MacKay. The structure of words and syllables: Evidence from errors in speech. *Cognitive Psychology*, 3:210–227, 1972.

[40] S. Oviatt. Predicting spoken disfluencies during human-computer interaction. *Computer Speech and Language*, 9:19–35, 1995.

[41] E. Shriberg. *Preliminaries to A Theory of Speech Disfluencies*. PhD thesis, University of California at Berkeley, 1994.

[42] H. H. Clark and T. Wasow. Repeating words in spontaneous speech. *Cognitive Psychology*, pages 201–242, 1998.

[43] R. Lickley. Missing disfluencies. In *Proceedings of International Congress of Phonetics Sciences*, pages 192–195, 1995.

[44] S. E. Brennan. How listeners compensate for disfluencies in spontaneous speech. *Journal of Memory and Language*, 44:274–296, 2001.

[45] J. E. Fox Tree. The effects of false starts and repetitions on the processing of subsequent words in spontaneous speech. *Journal of Memory and Language*, 34:709–738, 1995.

[46] L. Chen, M. Harper, and F. Quek. Gesture patterns during speech repairs. In *Proceedings of the International Conference on Multimodal Interfaces*, 2002.

[47] K. Manyhart. Age-dependent types and frequency of disfluencies. In *Proceedings of the Disfluency in Spontaneous Speech Workshop*, pages 45–48, 2003.

[48] H. Bortfeld, S. D. Leon, J. E. Bloom, M. F. Schober, and S. E. Brennan. Disfluency rates in conversation: Effects of age, relationship, topic, role and gender. *Language and Speech*, 2001.

[49] J. Bear, J. Dowding, and E. Shriberg. Integrating multiple knowledge sources for detecting and correction of repairs in human-computer dialog. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 56–63, 1992.

[50] E. Charniak and M. Johnson. Edit detection and parsing for transcribed speech. In *Proceedings of the North American Chapter of the Association for Computational Linguistics annual meeting*, pages 118–126, 2001.

[51] M. Johnson and E. Charniak. A TAG-based noisy channel model of speech repairs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2004.

[52] M. G. Core and L. K. Schubert. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 413–420, 1999.

[53] K. Zechner. *Automatic Summarization of Spoken Dialogues in Unrestricted Domains*. PhD thesis, Carnegie Mellon University, 2001.

[54] P. Lendvai, A. V. D. Bosch, and E. Krahmer. Memory-based disfluency chunking. In *Proceedings of the Disfluency in Spontaneous Speech Workshop*, pages 63–66, 2003.

[55] A. Stolcke and E. Shriberg. Statistical language modeling for speech disfluencies. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 1996.

[56] P. Heeman and J. Allen. Speech repairs, intonational phrases and discourse markers: Modeling speakers' utterances in spoken dialogue. *Computational Linguistics*, 25:527–571, 1999.

[57] W. Levelt and A. Cutler. Prosodic marking in speech repair. *Journal of Semantics*, 2:205–217, 1983.

[58] A. Cutler and D. R. Ladd, editors. *Prosody: Models and Measurement*, chapter Speakers? Conceptions of the Function of Prosody, pages 79–91. Heidelberg: Springer-Verla, 1983.

[59] R. Lickley, R. Shllcock, and E. Bard. How and when are disfluencies found? In *Proceedings of the European Conference on Speech Communication and Technology*, 1991.

[60] R. Lickley and E. Bard. When can listeners detect disfluency in spontaneous speech? *Language and Speech*, pages 203–226, 1998.

[61] D. Hindle. Deterministic parsing of syntactic nonfluencies. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 123–128, 1983.

[62] E. Shriberg. Phonetic consequences of speech disfluency. In *Proceedings of the International conference of Phonetics Sciences*, pages 619–622, 1999.

[63] R. Lickley. Juncture cues to disfluency. In *Proceedings of the International Conference on Spoken Language Processing*, 1996.

[64] G. Savova and J. Bachenko. Prosodic features of four types of disfluencies. In *Proceedings of the Disfluency in Spontaneous Speech Workshop*, pages 91–94, 2003.

[65] D. O'Shaughnessy. Analysis and automatic recognition of false starts in spontaneous speech. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, pages 724–727, 1993.

[66] M. Snover, B. Dorr, and R. Schwartz. A lexically-driven algorithm for disfluency detection. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, 2004.

[67] W. Levelt. Monitoring and self-repair in speech. *Cognition*, pages 41–104, 1983.

[68] A. Kai and S. Nakagawa. Investigation on unknown word processing and strategies for spontaneous speech understanding. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2095–2098, 1995.

[69] M. Goto, K. Itou, and S. Hayamizu. A real-time filled pause detection system for spontaneous speech recognition. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 227–230, 1999.

[70] M. H. Siu and M. Ostendorf. Modeling disfluencies in conversational speech. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, pages 386–389, 1996.

[71] S. Strassel. *Simple Metadata Annotation Specification V5.0.* Linguistic Data Consortium, 2003.

[72] DARPA Information Processing Technology Office. Effective, affordable, reusable speech-to-text (EARS). http://www.darpa.mil/ipto/programs/ears/, 2003.

[73] S. Strassel. *Simple Metadata Annotation Specification V6.2*. Linguistic Data Consortium, 2004.

[74] D. Hand. *Construction and Assessment of Classification Rules*. John Wiley and Sons, Chichester, 1997.

[75] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(6):1145–1159, 1997.

[76] R. O. Duda and P. E. Hart. *Pattern Recognition and Scene Analysis*. New York: John Wiley & Sons, 1973.

[77] M. Ostendorf and D. Hillard. Scoring structural mde: Towards more meaningful error rates. In *EARS Rich Transcription Workshop*, 2004.

[78] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. Ramana Rao Gadde, M. Plauché, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng. The SRI March 2000 Hub-5 conversational speech transcription system. In *Proceedings of NIST Speech Transcription Workshop*, College Park, MD, May 2000.

[79] L. Ferrer. Prosodic features for the switchboard database. Technical report, SRI International, 2002.

[80] K. Sonmez, E. Shriberg, L. Heck, and M. Weintraub. Modeling dynamic prosodic variation for speaker verification. In *Proceedings of the International Conference on Spoken Language Processing*, pages 3189–3192, 1998.

[81] L. Breiman, J. J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Pacific Grove, CA:Wadsworth and Brooks, 1984.

[82] W. Buntime and R. Caruana. *Introduction to IND version 2.1 and Recursive Partitioning*. NASA Ames Research Center, Moffett Field, CA, 1992.

[83] A. Stolcke and E. Shriberg. Automatic linguistic segmentation of conversational speech. In *Proceedings of the International Conference on Spoken Language Processing*, pages 1005–1008, 1996.

[84] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[85] C. Bishop. *Neural Networks for Pattern Recognition*. Cambridge University Press, Cambridge, UK, 1995.

[86] J. Kim. Automatic detection of sentence boundaries, disfluencies, and conversational fillers in spontaneous speech. Master's thesis, University of Washington, 2004.

[87] A. Sankar, L. Heck, and A. Stolcke. Acoustic modeling for the SRI Hub4 partitioned evaluation continuous speech recognition system. In *Proceedings of DARPA Speech Recognition Workshop*, pages 127–132, 1997.

[88] Y. Liu. Metadata extraction for rich transcription of speech. Technical report, Purdue University, Electrical and Computer Engineering Department, 2003.

[89] W. Wang and M. Harper. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources in language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 238–247, 2002.

[90] C. Chelba. *Exploiting Syntactic Structure for Natural Language Modeling*. PhD thesis, John Hopkins University, 1999.

[91] S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. Technical report, Harvard University, Computer Science Group, 1998.

[92] R. Knerser and H. Ney. Improved backing-off for N-gram language modeling. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, pages 181–184, 1995.

[93] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[94] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall, 1990.

[95] F. Jelinek. Self-organized language modeling for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*. Morgan Kaufman Publishers, 1990.

[96] P. A. Heeman. POS tagging versus classes in language modeling. In *Proceedings of the 6th Workshop on Very Large Corpus*, 1998.

[97] T. R. Niesler and P. C. Woodland. Variable-length category-based N-gram language model. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, pages 164–167, 1996.

[98] M. Johnson. Joint and conditional estimation of tagging and parsing models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2001.

[99] W. Wang. *Statistical Parsing and Language Modeling Based on Constraint Dependency Grammar*. PhD thesis, Purdue University, 2003.

[100] W. Wang, A. Stolcke, and M. Harper. The use of a linguistically motivated language model in conversational speech recognition. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2004.

[101] J. R. Bellegarda. A multi-span language modeling framework for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, 6:456–467, 1998.

[102] J. Peters. Compact maximum entropy language models. In *Proceedings of Automatic Speech Recognition and Understanding Workshop*, 1999.

[103] R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, Carnegie Mellon University, 1994.

[104] M. H. Siu and M. Ostendorf. Variable Ngrams and extensions for conversational speech language modeling. *IEEE Transactions on Speech and Audio Processing*, 8:63–75, 2000.

[105] P. F. Brown, V. J. D. Pietra, P. V. DeSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, pages 467–479, 1992.

[106] T. Brants. TnT a statistical part-of-speech tagger. In *Proceedings of the 6th Applied NLP Conference*, pages 224–231, 2000.

[107] http://www.ldc.upenn.edu.

[108] R. Florian and G. Ngai. Multidimensional transformational-based learning. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 1–8, July 2001.

[109] A. Stolcke. SRILM – An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904, 2002.

[110] N. V. Chawla, N. Japkowicz, and A. Kolcz. Workshop on learning from imbalanced datasets II, 20th International Conference on Machine Learning, August 2003.

[111] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, pages 321–357, 2002.

[112] M. Kubat and S. Matwin. Addressing the curse of imbalanced training sets. In *Proceedings of the International Conference on Machine Learning*, pages 179–186, 1997.

[113] J. Laurikkaka. Improving identification of difficult small classes by balancing class distribution. Technical report, Department of Computer and Information Science, University of Tampere, Finland, 2001.

[114] M. Kubat, R. Holte, and S. Matwin. Learning when negative examples abound. In *Proceedings of European Conference on Machine Learning*, pages 146–153, 1997.

[115] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–450, 2002.

[116] F. Provost and T. Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

[117] S. Lee. Noisy replication in skewed binary classification. *Computational Statistics and Data Analysis*, 34:165–191, 2000.

[118] P. Chan and S. Stolfo. Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 164–168, 1998.

[119] C. Ling and C. Li. Data mining for direct marketing problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 73–79, 1998.

[120] G. Weiss and F. Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Artificial Intelligence Research*, pages 315–354, 2003.

[121] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[122] B. Wrede and E. Shriberg. Spotting "hotspots" in meetings: Human judgments and prosodic cues. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 2805–2808, 2003.

[123] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[124] N. V. Chawla, T. E. Moore, L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, and L. Springer. Distributed learning with bagging-like performance. *Pattern Recognition Letters, Vol. 24 (1-3)*, pages 455–471, 2003.

[125] D. Drummond and R. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. In *Proceedings of ICML'03 Workshop on Learning from Imbalanced Datasets*, 2003.

[126] N. V. Chawla. C4.5 and imbalanced datasets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure. In *Proceedings of the ICML'03 Workshop on Class Imbalances*, 2003.

[127] F. Roli and J. Kittler, editors. *Multiple Classifier Systems*. Springer, 2002.

[128] F. Roli and J. Kittler, editors. *Multiple Classifier Systems*. Springer, 2001.

[129] G. Ngai and R. Florian. Transformation-based learning in the fast lane. In *Proceedings of the North American Chapter of the Association for Computational Linguistics annual meeting*, pages 40–47, June 2001.

[130] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–72, 1996.

[131] A. Ratnaparkhi. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996.

[132] K. Toutanova and C. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2000.

[133] K. Nigamy, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proceedings of IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[134] R. Koeling. Chunking with maximum entropy models. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 139–141, 2000.

[135] F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2002.

[136] S. Khudanpur and J. Wu. A maximum entropy language model integrating N-grams and topic dependencies for conversational speech recognition. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 1999.

[137] A. Borthwick. *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University, 1999.

[138] J. R. Curran and S. Clark. Language independent NER using a maximum entropy tagger. In *Proceedings of the Conference on Computational Natural Language Learning*, 2003.

[139] H. L. Chieu and H. T. Ng. Named entity recognition: A maximum entropy approach using global information. In *Proceedings of the International Conference on Computational Linguistics*, 2002.

[140] A. Suarez. A maximum entropy-based word sense disambiguation system. In *Proceedings of the International Conference on Computational Linguistics*, 2002.

[141] D. Klein and C. Manning. Conditional structure versus conditional estimation in NLP models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 9–16, 2002.

[142] S. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, 1999.

[143] L. Zhang. Maximum entropy toolkit. http://www.nlplab.cn/zhangle/maxent/toolkit.html, 2004.

[144] Y. Yang and J. P. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the International Conference on Machine Learning*, pages 412–420, 1997.

[145] T. E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74, 1993.

[146] J. Lafferty, A. McCallum, and F. Pereira. Conditional random field: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2001.

[147] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, 2003.

[148] A. McCallumn and W. Li. Early results for named entity recognition with conditional random fields. In *Proceedings of the Conference on Computational Natural Language Learning*, 2003.

[149] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, pages 329–336, 2004.

[150] A. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[151] J. Ajmera and C. Wooters. A robust speaker clustering algorithm. In *Proceedings of Automatic Speech Recognition and Understanding Workshop*, 2003.

[152] R. Lickley. *Detecting Disfluency in Spontaneous Speech*. PhD thesis, University of Edinburgh, 1994.

[153] R. C. Rose and G. Riccardi. Modeling disfluency and background events in asr for a natural language understanding task. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 1999.

[154] G. Fant. A four-parameter model of glottal flow. *STL-QPSR*, pages 1–13, 1985.

[155] G. Fant. The LF-model revisited: Transform and frequency domain analysis. *STL-QPSR*, pages 119–156, 1995.

[156] G. Fant. The voice source in connected speech. *Speech Communication*, 22:125–139, 1997.

[157] A. E. Rosenberg. The effect of glottal pulse shape on the quality of natural vowels. *Journal of the Acoustical Society of America*, 49:583–590, 1970.

[158] P. Boersma and D. Wennik. Praat, a system for doing phonetics by computer. http://www.praat.org, 1996.

[159] B. Blankenship. *The Time Course of Breathiness and Laryngealization in Vowels*. PhD thesis, UCLA, 1997.

[160] N. Duta and R. Schwartz. Error analysis of the BN and CTS results. EARS STT Workshop, St. Thomas, December, 2003.

[161] S. Coquoz. Broadcast news segmentation using MDE and STT information to improve speech recognition. Technical report, International Computer Science Institute, 2004.

[162] A. Stolcke et al. Speech-to-text research at SRI-ICSI-UW, 2003. http://www.nist.gov/speech/tests/rt/rt2003/spring/presentations/index.htm.

[163] L. Chen, Y. Liu, M. Harper, and E. Shriberg. Multimodal modal integration for sentence unit detection. In *Proceedings of the International Conference on Multimodal Interfaces*, 2004.

[164] A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. The ICSI meeting corpus. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2003.

[165] J. Ang, Y. Liu, and E. Shriberg. Automatic dialog act segmentation and classification in multiparty meetings. In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2005.

[166] E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey. The ICSI meeting recorder dialog act (MRDA) corpus. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, 2004.

[167] R. Dhillon, S. Bhagat, and H. Carvey et al. Meeting recorder project: Dialog act labeling guide. Technical report, International Computer Science Institute, 2004.

[168] D. Hillard, M. Ostendorf, A. Stolcke, Y. Liu, and E. Shriberg. Improving automatic sentence boundary detection with confusion networks. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting*, pages 69–72, 2004.

[169] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, pages 256–285, 1996.

[170] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth National Conference*, pages 148–156, 1996.

[171] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceedings of the International Conference on Machine Learning*, pages 124–133, 1999.

APPENDICES

APPENDICES

Appendix A: ADT Boosting For SU and IP Detection

In Chapter 7, bagging and ensemble techniques were explored to obtain more robust classifiers and thus a more reliable posterior probability of an event given the prosodic features. Bagging was used in our experiments largely because of its computational efficiency. In this section, we describe some other preliminary experiments on using boosting method for more robust classifiers. These experiments were placed in the appendix since they are not used in our full structural event detection system.

A.1 ADT Boosting Description

Freund and Schapire introduced AdaBoost (adaptive boosting) [169, 170] to improve classification performance by combining multiple weak learning algorithms, which has proven successful in many classification tasks. In boosting, each classifier is built based on the output of other classifiers, mostly by focusing on the samples for which they made incorrect decisions. The boosting algorithm is implemented by updating the weight of each sample in the training set. In contrast to bagging, boosting generates classifiers sequentially and thus cannot be implemented in parallel like bagging. It generates classifiers from different skewed distributions (due to the different weights for each sample used in each iteration); hence, this may affect our ability to combine boosting results, and also their combinations with the LM. The boosting algorithm was not used in our investigation of the imbalanced data set problem in Chapter 7 due to the above reasons.

In [171], Freund and Mason proposed an alternating decision tree (ADT) learning algorithm based on boosting that produces a single tree, which is a generalization of the classical decision tree. Figure A.1 shows an example of such an ADT tree. In each node, questions can be asked about different features. For example, Node 2 and 3 contain different decision questions, and yet they share one parent node. This is different from a classical decision tree, in which there can be only one question asked at each node. The tree built in this way is similar to that generated using the Bayes option in the IND package [82].

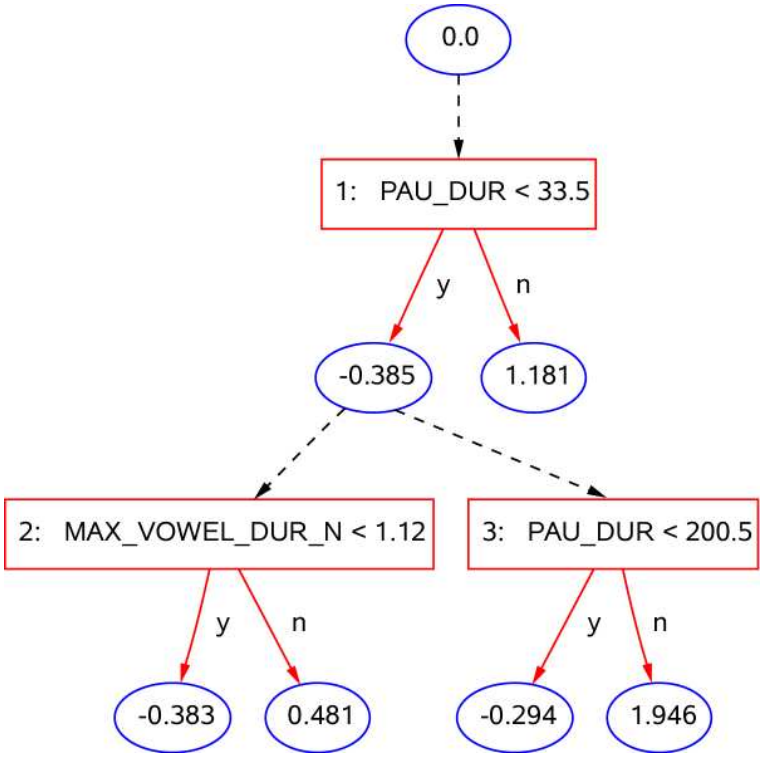

Fig. A.1. An example of an alternating decision tree (ADT).

A.2 Experimental Results

Since the ADT boosting algorithms runs faster than AdaBoosting, we applied this algorithm to SU boundary and IP detection tasks. Because the ADT boosting algorithm does not generate posterior probabilities of class membership for each test sample,[1] we only use this on a downsampled training and test set, and do not report results on the original test set either using the prosody model alone, or when combining the prosody model with the LM, both of which need posterior probabilities provided by the trees.

Table A.1
SU and IP detection results (classification error rate in %) using ADT learning algorithm and bagging. Training and testing were conducted using a downsampled training and testing set. Chance performance is 50%.

	Bagging	Boosting ADT
SU	14.30	14.8
IP	20.64	19.3

For this investigation, the data is the same as used in Section 7.3 for the study of the SU and IP detection tasks. The model is trained using a downsampled training set and tested on a downsampled test set. Experimental results using the ADT algorithm for the prosody model alone are shown in Table A.1. These results show that the ADT boosting algorithm improves performance for the IP task but not for the SU detection task, compared to the performance of bagging. This highlights the difference between the SU and IP tasks, suggesting that the metric of reducing classification errors used by the ADT learning algorithm may be better for the noisy IP task; whereas, information gain used in classical decision tree learning is more

[1] In future work, we will investigate methods for converting the score of the ADT learning algorithm to a posterior probability.

appropriate for the SU task. It is likely that the ADT learning algorithm is more robust to noisy data, and better at exploiting information from a small training set.

A.3 ADT Boosting Summary

In Chapter 7, we have shown some inherent differences between the SU and IP detection tasks; on the one hand they have different class distributions, on the other hand, they differ in the effectiveness of the features used in the tasks. The experiments on boosting show further that these two tasks are impacted differently by the learning algorithms. These are only preliminary results since they were conducted on a downsampled data set. Additional investigation on the generation of posterior probabilities from the ADT boosting algorithm, and thus evaluation on the real test set and in combination with the LM is needed.

Appendix B: Prosodic Features

All the abbreviated prosodic features that have appeared in this thesis are explained in this section. We do not include the whole prosodic feature set used in our structural event detection here. Details about each can be found in [79]. The subset of the features shown below also reveals how the prosodic features are correlated, being derived from the same raw features, with different binning or normalization methods applied. Note again that each feature is associated with and relative to a word w, as is indicated in the feature descriptions below.

1. Duration features

   PAU_DUR the duration of pause after word w_i

   TURN_F whether there is a speaker turn change after word w_i

   GEN gender of the speaker who uttered word w_i

   PREV_PAU_DUR the duration of pause before word w_i

LAST_RHYME_DUR_PH_bin binned value of [(the last rhyme duration of w_i) / (the number of phones in the last rhyme of w_i)]

LAST_RHYME_DUR_PH_ND_bin binned value of [LAST_RHYME_DUR_PH of w_i - (the speaker's mean)]

STR_RHYME_DUR_PH_bin binned value of [(the stressed rhyme duration of w_i) / (the number of phones in the stressed rhyme of w_i)]

LAST_VOW_DUR_Z_bin binned value of the last vowel duration of w_i (with variance normalization)

LAST_VOW_DUR_N_bin binned value of the last vowel duration of w_i (with mean normalization)

TURN_TIME_N (time so far in the current turn) / (the total duration of the current turn)

WORD_DUR the duration of word w_i

MAX_PHONE_DUR_Z, AVG_PHONE_DUR_N, MAX_PHONE_DUR_N, AVG_PHONE_DUR_ZSP, AVG_PHONE_DUR_NSP, MAX_PHONE_DUR_NSP, AVG_VOWEL_DUR_Z, AVG_VOWEL_DUR_N These duration features above are the average or maximum normalized phone duration in word w_i: _Z means variance normalization over all the speakers, _N means mean normalization over all the speakers, _ZSP means variance normalization over the current speaker, and _NSP means mean normalization over the current speaker

2. F0 features

PATTERN_WORD this feature consists of a sequence of 'f', 'uv', and 'r', representing a falling slope, an unvoiced region, and a rising slope in word w_i

PATTERN_BOUNDARY the last 'f' or 'r' in PATTERN_WORD for word w_i concatenated with the first slope tag for w_{i+1}

F0K_WRD_DIFF_LOLO_N log ratio between the minimum median filtered F0 value of w_i and w_{i+1}

F0K_WIN_DIFF_LOHI_N log ratio between the minimum median filtered F0 before the boundary and the maximum value after the boundary, both are within N frame window of w_i

F0K_DIFF_LAST_KBASELN the last good PWL fitted F0 value of w_i relative to the speaker's baseline F0

3. Energy features

ENERGY_WIN_DIFF_HIHI_N log ratio between the highest stylized energy value before and after w_i, both are within N frame window of the current boundary of w_i

ENERGY_PATTERN_BOUNDARY the last energy slope of w_i concatenated with the first energy slope of w_{i+1}

VITA

VITA

Education:

Purdue University: Ph.D candidate 2000-present

Electrical and Computer Engineering Department

Major Advisor: Mary P. Harper

Tsinghua University: B.A 1997, M.S 2000

Electrical Engineering Department

Selected Publications:

Journal Papers:

1. Yang Liu, Nitesh Chawla, Mary Harper, Elizabeth Shriberg, and Andreas Stolcke. "A Study in Machine Learning from Imbalanced Data for Sentence Boundary Detection in Speech". *Computer Speech and Language*, submitted.
2. Yang Liu, Mary Harper, Mike Johnson, and Leah Jamieson. "The Effect of Pruning and Compression on Graphical Representations of the Output of a Speech Recognizer". *Computer Speech and Language*, 2003.

Conference and Workshop Papers:

1. Jeremy Ang, Yang Liu, and Elizabeth Shriberg, "Automatic Dialog Act Segmentation and Classification in Multiparty Meetings", To appear in *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2005.
2. Yang Liu, Andreas Stolcke, Mary Harper, and Elizabeth Shriberg, "Comparing and Combining Generative and Posterior Probability Models: Some

Advances in Sentence Boundary Detection in Speech", In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

3. Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Dustin Hillard, Mari Ostendorf, Barbara Peskin, and Mary Harper, "The ICSI-SRI-UW Metadata Extraction System", In *Proceedings of the International Conference on Spoken Language Processing*, 2004.
4. Yang Liu, Elizabeth Shriberg, Andreas Stolcke, and Mary Harper, "Using Machine Learning to Cope with Imbalanced Classes in Natural Speech: Evidence from Sentence Boundary and Disfluency Detection", In *Proceedings of the International Conference on Spoken Language Processing*, 2004.
5. Lei Chen, Yang Liu, Mary Harper, and Elizabeth Shriberg, "Multimodal Model Integration for Sentence Unit Detection", In *Proceedings of the International Conference on Multimodal Interfaces*, 2004.
6. Yang Liu, Elizabeth Shriberg, and Andreas Stolcke, "Automatic Disfluency Identification in Conversational Speech Using Multiple Knowledge Sources", In *Proceedings of the European Conference on Speech Communication and Technology*, 2003.
7. Yang Liu, "Word Fragment Identification Using Acoustic-Prosodic Features in Conversational Speech", In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting, Student Workshop*, 2003.
8. Wen Wang, Yang Liu, and Mary Harper, "Rescoring Effectiveness of Language Models Using Different Levels of Knowledge and Their Integration", In *Proceedings of the International Conference of Acoustics, Speech, and Signal Processing*, 2002.

Professional Activities:

1. Student member of IEEE
2. Member of the Association for Computational Linguistics