# Automatic Detection of Sentence Boundaries, Disfluencies, and Conversational Fillers in Spontaneous Speech

Joungbum Kim

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

Master of Science in Electrical Engineering

University of Washington

2004

Program Authorized to Offer Degree: Electrical Engineering

University of Washington

Graduate School

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

Joungbum Kim

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

Committee Members:

_____

Mari Ostendorf

_____

Katrin Kirchhoff

Date: _____

University of Washington

Abstract

# Automatic Detection of Sentence Boundaries, Disfluencies, and Conversational Fillers in Spontaneous Speech

Joungbum Kim

Chair of Supervisory Committee:
Professor Mari Ostendorf
Electrical Engineering

Ongoing research on improving the performance of speech-to-text (STT) systems has the potential to provide high quality machine transcription of human speech in the future. However, even if such a perfect STT system were available, readability of transcripts of spontaneous speech as well as their usability in natural language processing systems are likely to be low, since such transcripts lack segmentations and since spontaneous speech contains disfluencies and conversational fillers.

In this work, we experiment with methods to automatically detect sentence boundaries, disfluencies, and conversational fillers in spontaneous speech transcripts. Our system has a two stage architecture in which sentence boundaries and locations of interruption in speech are predicted first and disfluent regions and conversational fillers are identified later. Decision trees trained with prosodic and lexical features and a language model are combined to categorize word boundaries, and rules learned through transformation-based learning are used to identify disfluent regions and conversational fillers. The research looks at different methods of model combination and different categorical representations of word boundaries as a cue to disfluent regions.

Experiments were conducted on the Switchboard corpus using a state-of-the-art STT system with comparison of results to hand transcriptions. The use of both prosodic and lexical information sources and an integrated detection approach gave the best results, with

relatively good performance on sentence boundary detection. The system also achieved a high degree of accuracy in conversational filler detection on hand transcripts, but word recognition errors in STT transcripts substantially degraded edit region and conversational filler detection performance.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Professor Mari Ostendorf without whose support, knowledge and guidance this thesis would never have been completed. The author also wishes to thank Sarah Schwarm for her contribution to this work.

## DEDICATION

To my parents.

# Chapter 1

# INTRODUCTION

Transcripts of spontaneous speech are often difficult to comprehend even without the challenges arising from word recognition errors introduced by imperfect automatic speech-to-text (STT) systems (Jones, Wolf, Gibson, Williams, Fedorenko, Reynolds & Zissman 2003). As can be seen in the following STT transcript of a conversation excerpt, such transcripts lack punctuation that indicates clausal or sentential boundaries, and they also usually contain a number of disfluencies that would not normally occur in written language or in read speech.

> yeah i know well actually um one airport i frequently fly through um interesting it it is a small airport um and **date** i mean we went **to** security it was a lot of it was actually elderly people um manning the security

In the above STT transcript excerpt, incorrectly recognized words are shown in boldface. A version of this excerpt is shown below, with corrected words in bold face and regions deleted by the recognizer in italics:

> yeah i know well actually um one airport i frequently fly through um *kind of* interesting it it is a small airport um and **they** i mean *when* we went **through** security it was a lot of it was actually elderly people um manning the security

The presence of STT system errors clearly affects the intelligibility of the transcript, but its readability remains low even after all the recognition errors have been corrected. This is because the transcript lacks punctuation and capitalization and contains conversational fillers such as filled pauses ("um") and discourse markers ("well" or "i mean") as well as edit

disfluencies like repeated words ("it it"), mid-sentence corrections of utterances ("it was a lot of it was"), and restarted utterances ("and they i mean when we went through"). Once we add punctuations and capitalizations to the transcript and remove fillers and disfluencies, the transcript becomes much more readable:

> Yeah, I know. One airport I frequently fly through, kind of interesting, it is a small airport, and when we went through security, it was actually elderly people manning the security.

The conversational fillers and edit disfluencies illustrated above are normal parts of spontaneous speech and easily handled by human listeners (Shriberg 1994). Filled pauses and discourse markers are often employed by speakers during a moment of hesitation or a change in discourse, and repeating words or altering utterances extemporaneously are perhaps unavoidable consequences of spontaneous speech. However, they generally decrease the readability of spontaneous speech transcripts because they typically interrupt the flow of the discourse (Jones et al. 2003). Furthermore, the existence of disfluencies makes transcripts of spontaneous speech ill-suited for most natural language processing (NLP) systems developed for text, because disfluencies are not modeled in those systems.

Similarly, the lack of any meaningful segmentation in the automatic transcripts of spontaneous speech makes them difficult to comprehend and problematic to use in NLP systems, most of which are designed to work on the sentential level. Therefore, detecting and removing edit disfluencies and locating sentential unit boundaries in the spontaneous speech transcripts can improve their readability and make them more suitable for NLP systems such as parsers or information extraction systems. Automatically annotating discourse markers and other conversational fillers is also likely to be useful, since they sometimes act as guideposts in the flow of conversation.

Previous work on disfluency and sentence boundary detection studied a number of different lexical and acoustic-prosodic cues that might identify speech disfluencies and sentence boundaries. Hindle proposed a system that relied on human-annotated "signals" of interruption in the utterance and a rule-governed deterministic parser to correct edit disfluencies (Hindle 1983). His approach to the edit disfluency detection problem was notable

as it was based on the assumption that occurrences of edit disfluencies could be identified by searching for locations of abrupt interruption in the speech signal. Nakatani & Hirschberg (1994), Shriberg, Bates & Stolcke (1997), and Shriberg (1999) also hypothesized that edit disfluencies are associated with distinct prosodic events and experimented with using prosodic information to detect such events. Stolcke, Shriberg, Bates, Ostendorf, Hakkani, Plauche, Tür & Lu (1998) introduced the use of language modeling in detecting edit disfluencies and sentence boundaries and investigated different methods of combining the language model with a prosody-based model.

We build on the knowledge gained through those studies and design a system to automatically detect edit disfluencies, fillers and sentence boundaries in wide-coverage conversational speech. The basic framework of our system on edit and filler detection is similar to Hindle's and to more recent work by Liu, Shriberg & Stolcke (2003), and our approach on sentence boundary detection is similar to work by Shriberg, Stolcke, Hakkani-Tür & Tür (2000). However, we take an integrated detection approach under the hypothesis that confusability and interdependency exist between disfluencies and sentence boundaries. We first utilize prosodic information and language models to locate sentence boundaries and sites of interruptions that indicate occurrences of disfluencies. Then we apply automatically trained rules to identify the words within those disfluencies and conversational fillers.

This thesis is organized as follows. In Chapter 2, we describe fillers, edit disfluencies and sentence boundaries in conversational speech in more detail, followed by a review of prior work. In Chapter 3, we introduce the experimental paradigm of our work. Chapter 4 discusses how prosodic and lexical information is used to locate sentence boundaries and disfluencies, and Chapter 5 documents our study on identifying the fillers and words in disfluencies. Finally we provide a summary and directions for future work in Chapter 6.

# Chapter 2

# BACKGROUND

## *2.1 Structural Metadata*

With the overall goal of improving usability of conversational speech transcripts in mind, we consider three main types of structural metadata: sentence-like units, conversational fillers and edit disfluencies. The motivation to choose these structures comes primarily from the availability of annotated conversational speech data from the Linguistic Data Consortium and standard scoring tools (NIST 2003*a*). Details of the annotation specification discussed in the following sections can be found in (Strassel 2003).

### *2.1.1 Sentence Units*

As illustrated in the conversation excerpt example of Section 1, spontaneous speech lacks the clear sentential boundaries of written text. As sentence fragments, ungrammatically constructed sentences, and backchannels ("uh-huh", "yeah") occur frequently in spontaneous speech, we instead attempt to detect *SUs* (variously referred to as sentence, semantic, and slash units). SUs are linguistic units roughly equivalent to sentences that are used to mark segmentation boundaries in conversational speech where utterances are often completed without forming "grammatical" sentences in the sense one thinks of with written text.

SUs can be sub-categorized into different types according to their discourse role. In our data, annotations distinguish statement, question, backchannel, and incomplete SUs. An SU is labeled as statement when it forms a complete declarative statement, either as a grammatically complete sentence or as a phrase or clause that functions as a standalone entity. Similarly, phrases, clauses or sentences that function as freestanding inquiry are labeled as questions, even if they were not actually in the form of a question. Backchannels refer to words or phrases used during a conversation to provide feedback or acknowledgement

Table 2.1: List of possible backchannels.

| |
|---|
| hm, hmm, huh, mm-hm, mm-hmm, oh, okay, OK, really, right, sure, yeah, yea, yep, yes, uh-huh |

to the dominant speaker, indicating that the conversation partner is listening. Due to difficulties in accurately labeling all backchannels, only the words listed in Table 2.1 were considered to be potential backchannels.

An SU was labeled as incomplete when the speaker trailed off and abandoned the SU without completing it or when the speaker was interrupted by another speaker before the SU could be finished. An example of an incomplete SU is shown in the following dialog where the end of an incomplete SU is marked by a slash followed by a dash (/-):

Speaker A: Yeah, but the thing about /-

Speaker B: No, see, you have to take inflation into account.

Speaker A: Oh, okay, I get it now.

In addition to SU types described above, SU-internal clause boundaries are also annotated in our data. However, those annotations were not used in this work.

### *2.1.2 Conversational Fillers*

In our study, conversational fillers or just fillers include filled pauses, discourse markers, and explicit editing terms. Filled pauses refer to hesitation sounds that speakers use to indicate uncertainties or stall the utterance. Discourse markers are words that speakers employ to signal change in conversation subject, to mark the start of a new utterance or topic, or to express their attitudes or feelings toward the current discourse. Examples of discourse markers are shown in boldface in the following dialogues:

Speaker A: Can you believe that?

Speaker B: **Now**, **see**, I'd never put those two together.

Table 2.2: List of filled pauses and discourse markers to be detected.

| | |
|---|---|
| Filled Pauses | ah, eh, er, uh, um |
| Discourse Markers | actually, anyway, basically, I mean, let's see, like, now, see, so, well, you know, you see |

Speaker A: The rest of it was all hog wash, but that was great.
Speaker B: **Well**, it seems like it would develop pride, **you know**, in people if, if it's in their own country.

A number of different hesitation sounds can act as filled pauses, and defining an all-inclusive set of English filled pauses is a problematic task. Deciding on a set of discourse markers is an even more complicated task because of the variety of ways in which discourse markers can be used. Therefore our system detects only a limited set of filled pauses and discourse markers listed in Table 2.2. Although filler detection is performed for a closed set, it appears that the fillers listed in Table 2.2 cover a large majority of cases (Strassel 2003).

An explicit editing term is a filler occurring within an edit disfluency, described further in the next section. For example, the discourse marker *I mean* serves as an explicit editing term in the following edit disfluency:

I didn't tell her that, *I mean*, I couldn't tell her that he was already gone.

### *2.1.3 Edit Disfluencies*

Although it is possible to categorize more narrowly, edit disfluencies largely encompass three separate phenomena that demonstrate different speaker behaviors: repetition, repair and restart (Shriberg 1994). A *repetition* occurs when a speaker repeats the most recently spoken portion of an utterance to hold off the flow of speech. A *repair* happens when the speaker attempts to correct a mistake that he or she just made. Finally, in a *restart*, the speaker abandons a current utterance completely and starts a new one. Examples of the different types of edit disfluencies are shown in Table 2.3.

Table 2.3: Examples of edit disfluencies.

| **Disfluency** | **Example** |
|---|---|
| **Repetition** | (I was) + **I was** very interested... |
| | (I was) + { uh} **I was** very interested... |
| **Repair** | (I was) + **she was** very interested... |
| | (I was) + { I mean } **she was** very... |
| **Restart** | (I was very) + Did you hear the news? |

Previous studies described a structure of edit disfluencies as an interval of self-correction or hesitation (Shriberg 1994, Nakatani & Hirschberg 1994). The first part of this structure is called the *reparandum*, a string of words that gets repeated or corrected. The reparandum is immediately followed by a non-lexical boundary event termed the *interruption point* (IP). In the edit disfluency examples shown in Table 2.3, reparanda are enclosed in parentheses, and IPs are represented by "+". The IP marks the point where the speaker interrupts a fluent utterance. In some cases, a filler follows the IP, explicitly indicating interruption in the flow of speech. For this reason, such a filler is called an explicit editing term. Furthermore, due to the unplanned and disruptive nature of edit disfluencies, word fragments frequently occur before an IP. The final part of the edit disfluency structure is called the *alteration*, which is a repetition or revised copy of the reparandum. In the case of a restart, the alteration is empty. In Table 2.3, the alterations are shown in boldface, and optional explicit editing terms are enclosed in braces.

In the case of complex edit disfluencies, where an edit disfluency occurs within a reparandum or an alteration of another disfluency, the overall structure of edit disfluencies becomes more complicated and the structures are organized into nested relationships. The following illustrates such an edit disfluency. In this example, the entire structure of an edit disfluency is enclosed in brackets, and the reparandum and the alteration are separated by the IP, denoted by the plus sign. As can be seen in the example, the reparandum or the alteration can be edit disfluencies themselves.

I guess if we all give five hours a week or five percent of our salary we could. . . uh, [ [ [ [ [ or just + or just gave ] + or ] + or ] + or ] [ just forced our kids + [ just to + force kids to ] ] serve for, uh, parents, I guess. . .

Due to higher cost of annotating complex edit disfluencies, the data we used was annotated with a flattened structure that treated them as simple disfluencies with multiple IPs (Strassel 2003). In other words, IPs within a complex disfluency were detected separately, and ranges of words in reparanda was determined to cover the deletable region of words for corresponding IPs. We will refer such a contiguous sequence of words in reparanda as an edit region.[1] The previous example of complex edit disfluencies is shown here again with the flattened structure of edit disfluencies. The edit regions are enclosed in braces, and IPs are represented by the plus sign.

I guess if we all give five hours a week or five percent of our salary we could. . . uh, { or just } + { or just gave } + { or } + { or } + or { just forced our kids } + { just to } + force kids to serve for, uh, parents, I guess. . .

### *2.2 Previous Work*

Some of the earliest work on automatic disfluency detection was reported by Hindle (1983). In that study, a deterministic parser and correction rules were used to clean up edit disfluencies. However it was not a truly automatic system as it relied on hand-annotated "edit signals" to locate IPs. Bear, Dowding & Shriberg (1992) explored pattern matching, parsing and acoustic cues and concluded that multiple sources of information are needed to detect edit disfluencies. A decision-tree-based system that took advantage of various acoustic and lexical features to detect IPs was developed by Nakatani & Hirschberg (1994).

Shriberg et al. (1997) applied machine prediction of IPs with decision trees to the broader Switchboard corpus by generating decision trees with a variety of prosodic features. Stolcke et al. (1998) then expanded the prosodic tree model of (Shriberg et al. 1997) with a hidden event language model (LM) to identify sentence boundaries, filled pauses and IPs in different

[1]This is sometimes refered to in the DARPA community as a "depod" (deletable part of a disfluency).

types of edit disfluencies in both human and STT transcripts. The hidden event LM used in their work adapted Hidden Markov Model (HMM) algorithms to an n-gram LM paradigm so that non-lexical events such as IPs and sentence boundaries could be represented as hidden states.

Liu et al. (2003) built on the framework developed in (Stolcke et al. 1998) and extended prosodic features and the hidden event language model to predict IPs in both human transcripts and STT system output. Their system also detected the onset of the reparandum by employing rule-based knowledge once IPs have been detected. Their system was able to improve chance IP detection performance (i.e. hypothesizing each word boundaries as non-IP boundaries) on human transcripts from 96.62% to 98.10%, where the chance accuracy is high because IPs are rare events. On predicting the onset of edit disfluencies, a recall rate of 61.45% and precision of 68.64% were obtained.

Edit disfluency detection systems that rely exclusively on word-based information have been presented by Heeman, Loken-Kim & Allen (1996) and Charniak & Johnson (2001). Heeman et al. took statistical language modeling approach to detect edit regions, discourse markers, intonational boundaries, and part-of-speech tags. Charniak and Johnson used lexical features derived from word identities, part-of-speech tags, and word fragment information and built a linear classifier to identify edit regions. Both of these efforts utilized the structure of edit disfluencies to detect edit regions. In the work of Heeman et al., this was done by modeling special tags that indicate onsets of reparanda and alterations. Charniak and Johnson incorporated the edit disfluency structure by generating features based on such a structure.

Some of earlier work on sentence boundary detection focused on disambiguating punctuation in text to locate sentence boundaries (Reynar & Ratnaparkhi 1997, Palmer & Hearst 1994). However, our study on the sentence boundary detection problem differs from such work since one of our goals is to automatically identify sentence boundaries in STT transcripts for which no punctuation or capitalization cues are available. Automatic detection of sentence boundaries using only machine-generated prosodic and lexical features has been studied by Stolcke et al. (1998) and Shriberg et al. (2000). Both of these work utilized a decision tree trained with prosodic features and a hidden event LM to identify

sentence boundaries in human and STT transcripts. Kim & Woodland (2001) also investigated the use of a combination model of a decision tree and a language model in automatic punctuation detection. Huang & Zweig (2002) built maximum-entropy based models to annotate punctuation in STT transcripts, and Christensen, Gotoh & Renals (2001) developed a finite-state model of words and prosody to mark punctuation.

Among this previous work, our approach to the disfluency detection problem is most similar to (Liu et al. 2003) in the sense that we also attempt to detect boundary events such as IPs first and use them as "signals" when identifying the reparandum in a later stage. The motivation to detect IPs first comes from the fact that, except for the location of an IP, speech is considered to be fluent and is likely to be free of any prosodic or lexical irregularity that can indicate the occurrence of an edit disfluency. Another similarity between the approach of (Liu et al. 2003) and ours can be found in the way mult iple knowledge sources were employed to predict IPs. Both systems utilized a decision tree trained with prosodic features and a hidden event language model for the IP detection task. However, we incorporate both lexical and prosodic features in the decision tree, and we used the transformation-based learning algorithm for the second stage of edit region detection. We also explore different methods of combining the hidden event language model and the decision tree model. In our work, posterior probabilities of boundary events obtained from the decision tree and hidden event language model are combined through linear interpolation, joint modeling, and HMM decoding approaches. These model combination strategies have been also explorered in (Stolcke et al. 1998) and (Shriberg et al. 2000).

Another key difference between our system and most previous work is the prediction target. Our system incorporates detecting word boundary events such as SUs and IPs, locating onsets of edit regions, and identifying filled pauses, discourse markers and explicit editing terms. We believe that this comprehensive detection scheme allows our system to better model dependencies between these events, which will lead to an improvement in the overall detection performance. Such a comprehensive approach is also being explored concurrently by researchers at SRI and ICSI (NIST 2003*b*), but in our case some events are modeled jointly under the hypothesis that it increases accuracy when these events are mutually confusable or interdependent. In addition, we study the effect of detecting multiple

types of IPs on boundary event prediction accuracy and edit region and filler detection performance.

# Chapter 3

# SYSTEM DESCRIPTION AND EXPERIMENTAL PARADIGM

## *3.1 Overall System Architecture*

As shown in Figure 3.1, our system detects disfluencies in a two-step process. First, for each word boundary in the given transcription, a decision tree (DT) trained with prosodic and lexical features and a hidden event LM (HE-LM) predict boundary events. Posterior probabilities of boundary events as estimated by the decision tree and hidden event LM are combined to make final boundary event predictions. In the second stage, rules learned via the transformation-based learning (TBL) algorithm are applied to the data containing predicted boundary events and lexical features extracted from the reference or STT transcripts. Edits and fillers are identified by labeling each word as being part of an edit region, a filled pause, a discourse marker, or an explicit editing term. It is possible for a word to be both a filler and part of an edit region, but these are infrequent and predicted only as a filler in our system.

## *3.2 Experimental Paradigm*

For training our system and its components, we used a subset of the Switchboard corpus (Godfrey, Holliman & McDaniel 1992). The Switchboard corpus is a collection of spontaneous conversational telephone speech (CTS) that average 6 minutes in length and reflect speakers of both sexes and major dialects of American English. The data set we used included 417 conversations that were hand-annotated by the Linguistic Data Consortium (LDC) for disfluencies and SUs according to guidelines detailed in (Strassel 2003), henceforth referred to as V5. We will refer to this data set as LDC1.3.

For development and testing of our system, we used hand transcripts and STT system output of 36 Switchboard and 36 Fisher conversations, where the Fisher conversations are

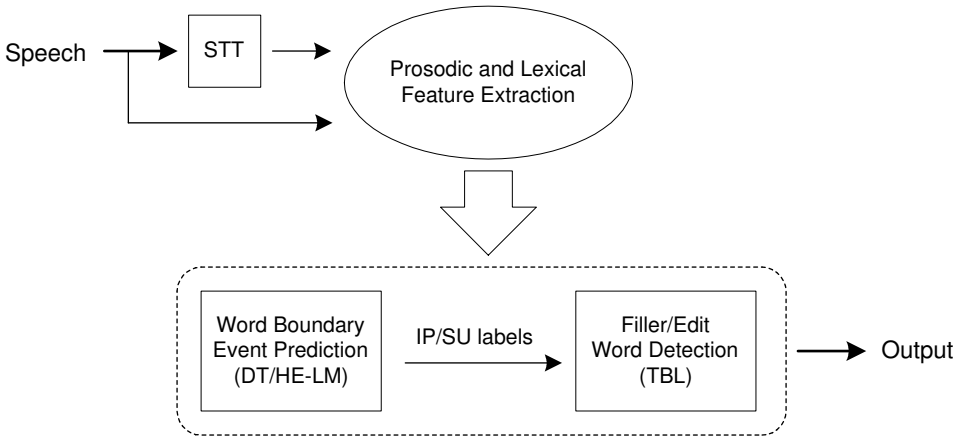


Figure 3.1: System Diagram

from a more recent data collection effort similar to the Switchboard corpus. Half of these conversations (18 Switchboard and 18 Fisher) were held out and used as development data, and the other 36 conversations were used as test data. The development and test data were annotated by the LDC using the V5 convention for the DARPA EARS Rich Transcription Fall 2003 evaluations (NIST 2003*a*).

The STT transcripts, used only in testing, were produced by automatically segmenting speech through the use of cross-channel event modeling (Liu & Kubala 2003) and recognizing it with the large vocabulary CTS recognition system developed by BBN Laboratories for Rich Transcription 2003 Spring evaluation (BBN 2003*b*). This is a state of the art system which obtains word error rates of 27% and 25% on the development and evaluation test sets, respectively. The STT hypotheses were force-aligned to generate word and phone times, which were used to generate prosodic features. Compound words recognized by the STT system were broken into individual words in a post-processing step using the phonetic dictionary utilized by the STT system and phone times from the forced alignment.

### *3.3 Evaluation Criteria*

The boundary event detection performance was evaluated by computing different scoring statistics. First, recall and precision of predictions were calculated for each boundary event

class to observe the quality of predictions for different boundary event classes. As a measure of overall performance, we computed word-based accuracy, which indicates percentage of correct predictions among all word boundaries. Finally, slot error rates (SER), which measure the ratio of all errors to reference events, were used to compare performances of different classifiers. The SER illustrated performance differences better than the word-based accuracy measure due to the large number of word boundaries that do not contain an SU or IP. However, the word-based measure was sometimes useful because this is what the decision tree optimizes and what the tree software reports.

Although performance evaluation is a trivial task for the reference transcripts, it becomes a much more challenging task for the STT transcripts since transcripts containing human annotations and the STT transcripts are different due to STT recognition errors. In order to align words and boundary event predictions in the STT transcripts to those in the reference transcripts, we utilized the capability of *rteval v2.3*, a scoring tool developed at the BBN Laboratories for the EARS Rich Transcription evaluation task (BBN 2003*a*).

Designed to find an alignment that achives the lowest word error rate of STT transcripts while minimizing error rates of SU and IP events as well as edit and filler detection, rteval utilizes the Levenshtein distance between hypothesized words and reference words using only the information about word identities and annotations. We created an alignment table of reference and STT transcripts by scoring the STT system output of evaluation and test data against the reference annotated transcripts and recording alignments made by rteval. This alignment table was then used to evaluate boundary event prediction performance on the STT transcripts.[1] For those cases where a reference word was not aligned to any word in the STT transcripts, the boundary event associated with the reference word, if any, was ignored. For example, if a word fragment was not recognized by the STT system and not aligned to any word in the STT trancript by rteval, the IP event following the word fragment was not used when computing error rates for IP detection.

Although rteval calculates error statistics of SU and IP detection, we decided not to use

[1]Since rteval attempts to minimize metadata (SU, IP, edit and filler) detection errors as well as STT word recoginition errors, it is possible to have slightly different alignment for the same STT transcripts if metadata hypotheses change. However, we believe any performance evaluation differences due to such alignment changes to be insignificant, since rteval puts more weight on minimizing word recognition errors.

rteval for boundary event prediction performance evaluation and instead used the alignment table for scoring. Scoring with the alignment table allowed us to compute separate error rates for different SU or IP types and gain more insight into the prediction performance of our systems. However, we used rteval to assess the performance of edit and filler detection since it provided sufficient error statistics for our analysis. In the conclusions of the subsequent chapters, we report the best case results for rteval as well as alternative scoring tools df-eval-v13 and su-eval-v15, since these are the accepted tools for the DARPA EARS Rich Transcription performance analysis.

It is difficult to determine statistical significances of metadata detection results evaluated by the scoring tools, since there is no standard established for computing them. However, we performed the sign test with P-value of 0.05 to verify statistical significances of performance differences between different metadata prediction models.

# Chapter 4

# DETECTING WORD BOUNDARY EVENTS

## *4.1 Word Boundary Events*

The word boundary events of interest in our work largely fall into two categories: those that mark the end of sentence-like units and those that signal interruptions associated with edit disfluencies and fillers. In view of the structural metadata described in Section 2.1, the first category of events would include the end of an SU. Similarly, the natural choice for the boundary event related to edit disfluencies would be the IP.

An end of an SU always coincides with the beginning of another since every word belongs to an SU by definition. Therefore, detecting the end of an SU is essentially equivalent to detecting the start of an SU, and our approach was to identify ends of SUs. In this section, we will use the term SU somewhat interchangeably to refer to both the sentential unit SU and the word boundary event indicating the end of an SU.

Since the annotation of the data used in our work includes labels for types of SUs, it is possible to further distinguish SU boundary events according to the type of the SU, namely statement, question, backchannel and incomplete. Although it is highly likely that each SU type will have acoustic and lexical characteristics unique to itself, we chose not to separate SU events into all four different categories because we suspected that we might not have enough data to train our models to perform such multi-class predictions optimally. Instead we considered just the incomplete SU to be separate from other SU types, since it represents unfinished utterances and might have significantly different characteristics. Furthermore, since restart type edit disfluencies can be also thought as abandoned utterances, high degree of confusability may exist between incomplete SUs and IPs.

Prior work has defined IPs to describe interruptions in the flow of speech within edit disfluencies. However, we use an extended definition of IPs and considered them to also

occur at word boundaries preceding fillers, since conversational fillers also often result in similar interruptions.

In summary, in the first stage of processing, our goal was to classify each word boundary as containing an SU, an incomplete SU (SU-inc), an IP, or none of these events (which will be refered to as the null event). Although SUs and IPs within edit disfluencies should never co-occur, it is possible to have both an SU and an IP at the same word boundary if a filler follows the end of an SU. We considered different options for handling these cases, including just predicting an SU and adding the IP in post-processing and explicit prediction of both.

### *4.2 Models*

In order to detect boundary events, we trained a CART-style decision tree (Breiman, Friedman, Olshen & Stone 1984) with various prosodic and lexical features. Decision trees are well-suited for this task because they provide a convenient way to integrate both symbolic and numerical features in prediction. Furthermore, it is easy to inspect the choice of features in a trained decision tree, which allows us to gain additional insight on the utilities of and the interactions between multiple information sources.

In addition to the decision tree model, we also employed a hidden event language model (HE-LM) to predict boundary events. The HE-LM is the same as a typical n-gram LM in all aspects except that it can be used to model non-lexical events in the n-gram context by counting special non-word tokens representing such events. In this way, the HE-LM effectively estimates the joint distribution $P(W, E)$ of words $W$ and events $E$. Once the model has been trained, a dynamic forward-backward programming algorithm can be used to calculate $P(E|W)$, or the posterior probability of an event given the word sequence (Stolcke et al. 1998, Stolcke & Shriberg 1996).

We also experimented with combining the decision tree model and the HE-LM in three different ways. In the first approach, which we will refer to as the joint tree model, the boundary event posterior probability from the HE-LM is jointly modeled with other features in the decision tree to make predictions about the boundary events. In the second approach, which we will refer to as the indepedent combination model, event posterior

probabilities from the decision tree and HE-LM are combined under the assumption that classification features used in the decision tree are mostly indepedent of the word sequence when conditioned on the boundary event. The decomposition of the combined boundary event posterior probability utilizing the conditional indepedence assumption is shown below.

Let $E$, $F$, and $W$ be n-length sequences of events, features, and words:

$$E = e_1, e_2, \ldots, e_n \qquad F = f_1, f_2, \ldots, f_n \qquad W = w_1, w_2, \ldots, w_n$$

Using the conditional independence assumption $P(f_i, W|e_i) = P(f_i|e_i)P(W|e_i)$,

$$\begin{aligned} P(e_i|f_i, W) &= \frac{P(f_i, W|e_i)P(e_i)}{P(f_i, W)} \\ &= \frac{P(W|e_i)P(f_i|e_i)P(e_i)}{P(f_i, W)} \\ &= \left(\frac{P(e_i|W)P(W)}{P(e_i)}\right)\left(\frac{P(e_i|f_i)P(f_i)}{P(e_i)}\right)\left(\frac{P(e_i)}{P(f_i, W)}\right) \\ &= \left(\frac{P(e_i|W)}{P(f_i|W)}\right)\left(\frac{P(e_i|f_i)P(f_i)}{P(e_i)}\right) \end{aligned}$$

Since $P(f_i|W)$ and $P(f_i)$ do not depend on $e_i$, they have no effect on boundary event prediction. Thus the following score was used when indepedently deciding on the most likely event for each word boundary:

$$(P_{LM}(e_i|W))^{\lambda}\left(\frac{P_{tree}(e_i|f_i)}{P(e_i)}\right)^{(1-\lambda)}$$

where the weighting factor $\lambda$ can be chosen empirically to maximize target performance, i.e. bias the prediction toward the more accurate model. Equivalently, we use a linear combination of log probabilities. The empirical weighting is important, because the models are trained on reference transcripts and word cues are less reliable in STT outputs.

Finally, in the third approach, which we will refer to as the integrated Hidden Markov Model (HMM), we attempt to jointly model decision tree features $f_i$, the word $w_i$ and the boundary event $e_i$ via the HMM. This approach augments that of the HE-LM by modeling $f_i$ as emissions from the HMM states represented by $w_i$ and $e_i$, with the emission likelihood $P(f_i|w_i, e_i)$ derived in the following way:

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

$$
\begin{aligned}
&= \frac{P(w_i)P(f_i|w_i)P(e_i|f_i,w_i)}{P(e_i,w_i)} \\
&= \frac{P(f_i|w_i)P(e_i|f_i,w_i)}{P(e_i|w_i)} \propto \frac{P(e_i|f_i,w_i)}{P(e_i|w_i)}
\end{aligned}
$$

With this framework, we can use the forward-backward algorithm to decode the model and determine boundary events. As an example, we define the forward probability $\alpha$ below:

$$
\begin{aligned}
\alpha_t(j) &= P(e_t = j, w_1^t, f_1^t) \\
&= \sum_{i=1}^{n} \alpha_{t-1}(i) P(e_t = j, w_t, f_t | e_{t-1} = i, w_{t-1}, f_{t-1}) \\
&= \sum_{i=1}^{n} \alpha_{t-1}(i) P(e_t = j, w_t | e_{t-1} = i, w_{t-1}) P(f_t | e_t = j) \\
&\propto \sum_{i=1}^{n} \alpha_{t-1}(i) P_{LM}(e_t = j, w_t | e_{t-1} = i, w_{t-1})^{\lambda} \left( \frac{P_{tree}(e_t = j | f_t)}{P(e_t = j)} \right)^{1-\lambda}
\end{aligned}
$$

Similarly to the indepedent combination model, a weighting factor $\lambda$ can be utilized to introduce desired bias to the combination model.

The joint tree model has the advantage that the (possibly) complex interaction between lexical and prosodic cues can be captured. However, since the data available for training the model is only on reference transcriptions, it tends to favor lexical cues, which become less reliable in STT output. In the independent combination model and joint HMM approaches, the relative weighting of the two knowledge sources can be estimated on the development test set for STT output, so it is possible for prosodic cues to be given a higher weight.

### *4.3 Boundary Event Detection with Decision Trees*

Decision trees to predict boundary events were trained and tested using the IND system developed by NASA (Buntine & Caruana 1991). All decision trees were pruned by ten-fold cross validation.

For each word boundary[1], a variety of lexical and prosodic features were generated

[1]For word fragments, we used their full forms instead of the partial representation noted by human

for the localized window of two words preceding and one word following the boundary. The prosodic features were generated to capture phenomena identified to be associated with boundary events of interest such as existence of silences, preboundary lengthening of phones, or different intonation patterns. The features included duration-based features, fundamental frequency (F0) and energy information and speaker turn markers[2] described in Section 4.3.1. On the other hand, lexical features were designed to capture syntactic characteristics found in edit regions or near sentence boundaries such as word repetitions or existence of pronouns. Section 4.3.2 describes them in more detail. Section 4.3.3 discusses our effort to select a more useful group of prosodic features, and boundary event detection results are reported in Section 4.3.4.

### *4.3.1 Prosodic Features*

**Duration-based features** included word and rhyme durations, rhyme duration differences between two neighboring words, and silence duration following a word. In our work, a rhyme was defined to contain the final vowel of a word and any consonants following the final vowel. Motivation for using these features comes from previous work on sentence boundary and disfluency detection, in which preboundary lengthening of vowels and pauses were frequently observed near sentence boundaries and IPs (Bear et al. 1992, Nakatani & Hirschberg 1994, Shriberg 1999, Shriberg et al. 2000).

The duration-based features were calculated from phonetic alignments obtained from the STT system. Word and rhyme durations were normalized by phone duration statistics obtained from phonetic alignments of the LDC1.3 and TB3-mapped data sets in two different ways. In one normalization method, duration $d_i$ of each phone of type $p(i)$ was normalized by the following equation:

---

annotators. Although such processing has no effect on prosodic feature generation, it could improve the quality of lexical features as irregular orthographic representations of word fragments make them difficult to model correctly. Since the STT system used in our work either fails to recognize word fragments or recognizes them as full words, using the full forms of fragments when training models may yield more accurate models.

[2]Even though speaker turn information is not strictly prosodic, we group it with the prosodic features since it is generated from acoustic information.

$$\bar{d}_i = \frac{d_i - \mu_{p(i)}}{\sigma_{p(i)}}$$

where the mean $\mu_p$ and standard deviation $\sigma_p$ of phone $p$ were calculated over all available training data. The rhyme or word duration then were computed by taking an arithmetic mean of normalized durations of the phones in the word or the rhyme.

In the second normalization approach, the word or rhyme duration was normalized by the sum of mean phone durations as follows:

$$\bar{D} = \frac{\sum_i d_i}{\sum_i \mu_{p(i)}}$$

We also generated **F0 and energy features** since research has shown that F0 and energy information may be useful for sentence boundary and disfluency detection (Levelt & Cutler 1983, Bear et al. 1992, Shriberg et al. 2000). The F0 features were generated by an autocorrelation-based pitch tracker included in the Entropic System ESPS/Waves package. Using the F0 stylization tool developed in (Sönmez, Shriberg, Heck & Weintraub 1998), we post-processed the F0 information to obtain stylized F0 contours. All F0 features were computed using the F0 stylizations in order to avoid noisy estimates. The ESPS/Waves package was also used to compute frame-level root mean square energy of the speech signal. In order to avoid non-speech signal reflected in energy features, we only used energy estimates of those frames identified to be within voiced regions according to the F0 stylization. The stylization tool also computed probability of F0 halving and doubling error for each frame from a lognormal tied mixture model of F0. Those frames with more than ten percent estimated chance of F0 doubling or halving were not used in F0 feature computation.

The F0 features generated included minimum, mean, and maximum value of stylized F0 contour over a word, slope of F0 contour at the end of a word, and differences in F0 statistics and F0 values at the start and end of two neighboring words at the candidate boundary. Similarly, energy features were minimum, mean and maximum energy value of a word and its rhyme. Both F0 and energy features were normalized by mean F0 and energy values calculated over each conversation side. Since all conversations in our data had only

two sides and one speaker per side, the F0 and energy features were effectively normalized for speaker differences.

**Speaker turn features** were also created for decision tree training since speaker turn boundaries coincide imply SU boundaries in our data. Determining speaker turns in our conversational telephone speech data was a more difficult problem than our data's two-channel, single-speaker-per-channel setup would suggest, because turn-taking was not moderated and it is not always easy to determine automatically which speaker has the floor. There were many instances of two speakers talking at the same time or one speaker giving comments while the other speaker continuing to talk. Such interactions between speakers led to difficulties in determining turn boundaries through speaker tracking. In order to avoid having a SU or an edit region split between two turns, we instead hypothesized turn boundaries at locations with silence durations of more than four seconds, which was the duration chosen through an empirical study of our data.

The turn boundary feature had following eight values to capture turn boundary information as well as speaker overlaps:

- 0: Not a turn boundary, no speaker overlap.
- OV: Speaker overlap exists.
- S: Start of a turn.
- SOV: Start of a turn with speaker overlap.
- E: End of a turn.
- OVE: End of a turn with speaker overlap.
- SE: Turn consists of a single word.
- SOVE: Turn consists of a single word and speaker overlap exists.

In addition to the turn boundary feature above, we generated another turn-based feature that indicated the ordinal position of a word in a turn.

### *4.3.2 Lexical Features*

Lexical features consisted of POS tag groups, word and POS tag pattern matches, and a flag indicating existence of filler words to the right of the current word boundary. POS tags can be useful for both SU detection since certain POS tags such as pronouns are highly indicative of sentence boundaries. Furthermore, POS tags can also give evidence to syntactic irregularities that may be associated with edit disfluencies. The pattern match features were designed to capture word repetitions often seen across IPs. The filler indicator feature was used to reflect word identity information, which is paricularly relevant for detection of IPs before fillers due to the use of closed set of fillers in our work.

The POS tag features were produced by first predicting the tags with Ratnaparkhi's Maximum Entropy Tagger (Ratnaparkhi 1996) and then clustered by hand into a smaller number of groups based on the syntactic roles that they serve as illustrated in Table 4.1. The clustering was performed to speed up decision tree training as well as to reduce the impact of tagger errors. Although the Maximum Entropy Tagger was originally designed to tag sentences with correct capitalization and punctuation, we used it to tag unicased text with no punctuation and with hypothesized turn boundaries utilized as sentence boundaries, since detecting sentence boundary was our goal and since the STT output lacked capitalization or punctuation. In order to improve tagging accuracy with such data, the tagger's models were also trained with similar unicased text from the TB3-mapped set.

Word pattern match features were generated by comparing words over the range of up to four words across the word boundary in consideration. Grouped POS tags were compared in a similar way, but the range was limited to at most two tags across the boundary since a wider comparison range would have resulted in far more matches than would be useful due to the low number of POS tag groups. When words known to be identified frequently as fillers existed after the boundary, they were skipped and the range of pattern matching was extended accordingly. An example of matched grouped POS tags are shown in Table 4.2. In the example, the grouped pos tags of "there are" are matched to those of "it's" while the grouped POS tags of "you know" are skipped because they are frequent filler words. The pattern match feature distinguished between exact match and partial match and identified

Table 4.1: Clustering part-of-speech tags.

| POS Group | POS Tags |
|---|---|
| NN | NN, NNP, NNPS, NNS, POS, GW, FW |
| CD | CD |
| AJ | JJ, JJR, JJS |
| AD | RB, RBR, RBS |
| VM | VBG, VBN |
| VL | VB, VBD, VBP, VBZ, BES |
| MD | MD, HVS |
| M1 | PP, PRP, WP, EX |
| M2 | PP$, PRP$, WP$, DT |
| M2R | IN, WDT, WRB |
| M2RS | RP |
| CC | CC |
| TO | TO |
| XX | UNK, XX |
| UH | UH |

Table 4.2: Example of grouped POS tag pattern match.

| Word | But | there | are | *you* | *know* | it's | like |
|---|---|---|---|---|---|---|---|
| POS | CC | EX | VBP | *PRP* | *VBP* | PRP+BES | IN |
| Grouped POS | M2 | **M1** | **VL** | *M1* | *M1* | **M1+VL** | M2 |

the number of words involved in the match.

Another useful cue for boundary event detection is the existence of word fragments (Bear et al. 1992, Nakatani & Hirschberg 1994). Since word fragments occur when the speaker cuts short the word being spoken, they are highly indicative of IPs. However currently available STT systems do not recognize word fragments reliably, and the STT system we used does not recognize them at all. As our goal is to build an automatic detection system, we chose not to use any features related to word fragments.

In addition, posterior probabilities of boundary events from the HE-LM was used as a lexical feature. However, we will defer the discussion of the performance of the decision tree trained with this feature until Section 4.5 where the study of different model combination approaches is presented.

### *4.3.3 Prosodic Feature Selection*

The greedy nature of the CART learning algorithm means that it will automatically choose and utilize the most imformative features. However, that very greediness of the algorithm can also lead to suboptimal performance when redundant features or features that do not contribute to the classfication of data are included in training, because such features may be chosen to minimize entropy locally and end up hurting overall performance ultimately. Since we considered a large number of prosodic features with varying degrees of utilities and redundancy, we attempted to determine the most effective subset of prosodic features by selecting only the prosodic features whose exclusion from the training process led to a decrease in boundary event detection accuracy on the reference transcripts of development data. More specifically, for each prosodic feature $x$, $P_{\bar{x}}$, the correct percentage of events

Table 4.3: Word-based accuracies for decision trees trained with selected and full set of prosodic features, on reference transcripts.

| Lexical Features Used | Development | | Test | |
|---|---|---|---|---|
| | Selected | All | Selected | All |
| Filler | 84.5 | 85.7 | 86.0 | 86.1 |
| POS, Match, Filler | 89.5 | 89.0 | 89.9 | 89.9 |
| LM, POS, Match, Filler | 91.1 | 90.9 | 91.7 | 91.7 |

detected by the decision tree trained without $x$ was compared to $P_{all}$, the correct percentage of events detected by the tree trained with all available prosodic features, and the prosodic feature was chosen to be used for training only if $P_{\bar{x}}$ was less than $P_{all}$.

Our experiments also indicated that inclusion or exclusion of different types of lexical features had an effect on the selection of prosodic features. In order to evaluate utilities of different lexical features through detection performances of the decision trees, we selected different sets of prosodic features for three different lexical feature groups. Table 4.3 shows detection performance comparisions between selected and full sets of prosodic features chosen for those lexical feature groups. In the table, "POS" stands for POS group features, "Match" for pattern match related features, "Filler" for the indicator of potential filler words, and "LM" for posterior probabilities of boundary events estimated by the HE-LM.

As can be expected, inclusion of more lexical features resulted in fewer numbers of selected prosodic features. Although different groups of prosodic features were selected for different lexical feature groups, some prosodic features were consistently chosen. Examples of these include rhyme duration differences across the word boundary, minimum energy over and slope at the end of the word preceding the word boundary, F0 value at the start of and mean energy over the rhyme of the previous word (the word to the left of the word preceding the word boundary), silence durations, and speaker turn features.

The selection algorithm actually resulted in *degradation* in performance when only a single lexical feature was used. In other cases, using a smaller set of prosodic features

yielded improvement on the development set but no significant difference on the test set. We believe the lack of improvements on the test data can be explained by the relatively simple nature of our leave-one-out feature selection algorithm as well as overoptimization on the development data. Even though the prosodic feature selection did not lead to clear improvements in performance, we utilize the set of selected prosodic features in our study since it considerably speeds up training decision trees.

### *4.3.4 Results and Discussion*

Several decision trees with different combinations of feature groups were trained to assess the usefulness of different knowledge sources for boundary event detection. The tree was then used to predict the boundary events on the test data. The results of the experiment on the reference transcripts are presented in Table 4.4, and the results on the STT transcripts are shown in Table 4.5. On the reference transcripts, the chance performance, or the correct percentage of predicted boundary events if all word boundaries were classified as the null event, was 76.49%. The chance performance on the STT transcripts was 77.89%. This higher chance performance on the STT transcripts reflects the fact that the STT system was more likely to fail to recognize those words associated with boundary events, such as filled pauses, word fragments or backchannels.

The decision tree trained with only prosodic features showed little difference in recall rates of boundary events between reference and STT transcripts, although precisions were higher on the reference transcipt. This is not surprising since prosodic features are far more robust to STT word recognition errors. However, since the decision tree was trained with prosodic features that utilized correct word hypotheses and time alignment, precision and overall accuracy were worse on the STT transcripts. The performance degradation caused by STT word errors become more prominent for the trees trained with lexical features.

As can be expected, addition of lexical features such as indicators of potential filler words and pattern match features that tend to identify occurrence of IPs improved IP detection but had little effect on SU detection. These features generally slightly decreased the recall rate of SUs but improved the precision. Interestingly, we observed the opposite

Table 4.4: Result of decision tree experiments on boundary event prediction, on reference transcripts.

| Features | SU | | SU-inc | | IP | | Word-based |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Recall | Precision | Accuracy |
| Prosody Only | 51.79 | 74.14 | 0 | - | 11.95 | 55.80 | 83.42 |
| Prosody, Filler | 51.71 | 74.41 | 0 | - | 42.03 | 79.26 | 86.03 |
| Prosody, POS, Filler | 70.93 | 79.15 | 25.00 | 63.57 | 61.26 | 77.09 | 89.80 |
| Prosody, POS, Match, Filler | 69.84 | 79.31 | 23.48 | 63.64 | 64.18 | 78.46 | 89.93 |

Table 4.5: Result of decision tree experiments on boundary event prediction, on STT transcripts.

| Features | SU | | SU-inc | | IP | | Word-based |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Recall | Precision | Accuracy |
| Prosody Only | 51.85 | 70.22 | 0 | - | 11.43 | 47.99 | 83.74 |
| Prosody, Filler | 53.51 | 70.06 | 0 | - | 34.54 | 64.13 | 85.32 |
| Prosody, POS, Filler | 68.58 | 73.42 | 21.68 | 47.50 | 49.54 | 62.21 | 87.43 |
| Prosody, POS, Match, Filler | 68.25 | 73.97 | 20.15 | 43.44 | 52.77 | 64.91 | 87.69 |

Table 4.6: Recall rates of SU events for different SU types.

| Features | Statement | | Question | | Backchannel | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| Prosody Only | 40.47 | 41.11 | 70.75 | 73.12 | 76.39 | 76.22 |
| Prosody, Filler | 41.15 | 43.69 | 69.36 | 72.83 | 74.88 | 76.13 |
| Prosody, POS, Filler | 65.11 | 62.22 | 70.47 | 73.12 | 88.38 | 88.02 |
| Prosody, POS, Match, Filler | 64.01 | 61.87 | 69.36 | 73.12 | 87.26 | 87.76 |

trend - SU recall increasing and SU precision decreasing - on the STT transcripts when the decision tree was using the filler indicator feature in addition to prosodic features. To explain this phenomenon, we computed the recall rates for different types of SUs separately. As illustrated in Table 4.6, the filler indicator feature generally improved statement type SU recall while hurting question and backchannel type SU detection. While this effect was true for both types of transcripts, degradation of detection performance for question and backchannel type SUs was much less pronounced on the STT transcript, leading to the improvement in overall SU recall rate. This suggests that those fillers correctly recognized by the STT system may be more likely to aid statement type SU detection without hurting detection of other types of SUs. The fact that the filler indicator feature only helped statement type SU detection might also indicate that speakers are less likely to employ conversational fillers at the start of an SU after a question or backchannel word, but further study is needed to confirm this. Addition of POS tag features substantially increased recall rates of statement and backchannel SUs, which demonstrates the importance of syntactic information in SU detection.

Table 4.6 also shows that the effectiveness of prosodic information in SU detection varies with the SU type. While the decision tree failed to detect more than half of statement type SUs with only prosodic features, it achieves far more success with backchannel and question

Table 4.7: Error breakdown for SU-inc events with the prosody-only tree.

| Features | Null | SU | SU-inc | IP |
|---|---|---|---|---|
| Reference | 10.06 | 88.41 | 0 | 1.52 |
| STT | 11.79 | 86.31 | 0 | 1.90 |

type SU detection. We believe this may be explained by the fact that backchannels are usually followed by long silences and that questions often result in speaker turn changes. Moreover, it is also possible that question type SUs have clearer F0 markings that better differentiate them from other word boundaries. The relatively small improvement in question type SU recall rate gained after inclusion of POS tag features further demonstrates the effectiveness of prosodic feature in question type SU detection.

One SU type missing in Table 4.6 is the incomplete SU. As shown in Tables 4.4 and 4.5, detection of this boundary event was particularly poor. However, as shown in Table 4.7, the majority of SU-inc events were classified as SU events by the decision tree trained with only prosodic features. This result hints that overall boundary event detection performance may be improved if SU-inc events are considered as just SU events. In Section 4.6.1, we will discuss experiments on merging SU-inc and SU categories.

Compared to detecting SU events, IP detection proved to be a more difficult task. Not only were the recall rates of IP events lower than that of SUs on the reference transcripts, but performance degradation was also much more severe when the STT transcripts were used.

As can be seen in Table 4.8, in which recall rates of different types of IPs are reported, IPs following edit regions are the most difficult to detect, and we essentially fail to detect this type of IPs when only the prosodic features were used. The detection performance was better for IPs occuring before fillers under the same condition, suggesting that word boundaries before fillers may be slightly more prosodically distinctive than word boundaries after edit regions. The filler indicator feature dramatically improved detection of IPs before fillers on reference transcript. However, it also explains more significant decrease in detection

Table 4.8: Recall rates of IP events for different IP types.

| Features | After Edit | | Between Edit&Filler | | Before Filler | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| Prosody Only | 2.69 | 2.17 | 21.63 | 20.47 | 18.89 | 17.02 |
| Prosody, Filler | 0.41 | 1.63 | 79.62 | 60.23 | 74.81 | 55.91 |
| Prosody, POS, Filler | 42.90 | 36.84 | 78.06 | 63.78 | 75.65 | 56.81 |
| Prosody, POS, Match, Filler | 48.69 | 44.74 | 78.68 | 64.96 | 76.23 | 56.63 |

accuracy for IP events on the STT transcripts. The word recognition errors in the STT transcripts make this feature much less reliable, resulting in lower detection performance.

The POS tag and pattern match features had much more pronounced effect on IPs associated with edit disfluencies than IPs that exist before fillers. This result indicates that syntactic structure and other lexical irregularities may be important source of information in detecting edit regions.

Although inclusion of more lexical features generally improve recall rates of different IP types, the recall rate of IPs occuring between edit regions and fillers on the reference transcripts actually decreased when POS tags were used in addition to prosodic and filler indicator features. We suspect that this result may be attributed to the substantially higher recall rate of SU events when POS tag features are utilized during event classification. As SU detection improves due to POS tag features, that might cause more word boundaries between edit regions and fillers to be misclassified as SU events. However, since we did not observe similar behavior on the STT transcripts or for IPs before fillers, additional study would be needed to confirm our hypothesis.

Another interesting trend noticeable in Table 4.8 is that the recall rate is the highest for those IPs between edit regions and fillers. Since this finding also applies to the case when no lexical features were used in event detection, it could imply that the “interruptions” in

Table 4.9: Recall rates for IPs associated with different fillers, on reference transcripts.

| Filler Type | # in Test Data | Prosody Only | Prosody, Filler | Prosody, POS, Filler | Prosody, POS Match, Filler |
|---|---|---|---|---|---|
| Filled Pauses | 800 | 33.0 | 96.8 | 93.4 | 93.4 |
| You know | 310 | 5.5 | 93.2 | 91.3 | 91.3 |
| Like | 198 | 2.5 | 0 | 14.6 | 16.2 |
| I mean | 62 | 4.3 | 65.2 | 78.3 | 82.6 |
| Well | 52 | 3.8 | 65.4 | 48.1 | 50 |

the flow of speech are more evident when edit regions are followed by fillers.

Detection performance of IPs occuring before fillers varied widely depending on the type of the filler as illustared in Table 4.9 which summarizes recall rates for the four most frequently occuring filler types[3] under different decision tree configurations. IPs before filled pauses and the discourse marker "you know" turned out to be the easiest to detect, and the filler indicator feature was the most useful in these cases. However, recall rates of IPs before the discourse markers "like" was considerably lower than other fillers. This result is not very surprising in light of the fact that even human annotators sometimes have difficulty correctly identifying "like" as discourse markers (Strassel 2003).

The pattern match features had no effect on the recall rates of IPs before filled pauses and "you know", but they were useful for other fillers. Inclusion of POS tag features resulted in a large drop in recall rates for IPs before "well" as well as a slight decrease for IPs before filled pauses and "you know". We suspect this can be attributed to increased recall rate of SU events when POS tag features are used. Since speakers usually employ "Well" at the start of SUs, IPs before "well" are most likely to be affected by changes in SU detection performance.

As discussed in Section 4.3.2, we decided not to use any word fragment related features

[3]These six fillers represented approximately 95% of all fillers in the test set.

in decision tree training due to low feasibility of automatic detection of fragments although word fragments are strongly associated with edit disfluencies. In order to gain more insight into how word fragments affect IP detection performance, we compared recall rates of the IPs occuring after word frements to that of other IPs that follow edit regions. As shown in Table 4.10, we had far less success in detecting the IPs after fragments than the IPs following other edit regions. Furthermore, we observed larger differences in recall rates of IPs following fragments versus other edit regions when lexical features were used, and pattern match features actually decreased the recall rate of fragment related IPs on the reference transcipt.

Although the lexical features were also less effective for fragment-related IPs on the STT transcripts, the performance differences were not as severe. When all available lexical features were used, the recall rate of IPs following fragments on the STT transcripts was close to the same rate on the reference transcripts, which was not the case for other kinds of IPs. We believe this phenomenon can be explained by the fact that the STT system we used either recognize word fragments as full words or not at all. Since word fragments are poorly modeled in our lexical features due to their irregular spelling, the lexical features are likely to be more useful when the word fragments are actually recognized as full words. This suggests that in the absence of explicit indicators of word fragments, it may be better to model them as full words instead of their partial word forms.

### *4.4 Boundary Event Detection with Hidden-event Language Model*

We used the SRI Language Modeling Toolkit (SRILM) (Stolcke 2002) to train a trigram open-vocabulary language model with Kneser-Ney discounting (Kneser & Ney 1995) on data that included boundary events SU, SU-inc, and IP as special non-word tokens[4]. When training the language model, we treated word fragments as full words and used their complete orthographic representations as noted by human transcribers. The word fragments in the development and test data were used in their partial word forms and considered as unknown words. Posterior probabilities of boundary events for every word boundary were

[4]The null event was implicitly modeled by the absence of other boundary events.

Table 4.10: Recall rates of IPs after fragments vs other edit regions.

| Features | After Fragment | | Other Edits | |
|---|---|---|---|---|
| | Reference | STT | Reference | STT |
| Prosody Only | 3.8 | 1.2 | 7.9 | 7.5 |
| Prosody, Filler | 12.4 | 7.4 | 18.7 | 16.2 |
| Prosody, POS, Filler | 29.3 | 21.1 | 59.4 | 48.6 |
| Prosody, POS, Match, Filler | 25.5 | 23.4 | 67.9 | 56.2 |

estimated by utilizing SRILM's capability for predicting hidden events through forward-backward dynamic programming.

The results of boundary event detection with the HE-LM are summarized in Table 4.11. As a comparison, the boundary event detection performance of a decision tree trained with prosodic and all available lexical features are also shown in the same table. The HE-LM generally outperformed the decision tree in SU detection, but the decision tree had higher recall on IP detection. Probably due to the use of prosodic cues, the decision tree suffered slightly smaller degree of performance degradation going from the reference to STT transcripts than the HE-LM. The fact that the HE-LM showed better SU detection performance suggests that word identity information is important in SU detection, which is not surprising since speakers very often start sentences with pronouns, fillers or conjunctions during a conversation. Although this information is also used in decision tree via POS tag and filler indicator features, the HE-LM can take advantage of longer lexical context through trigram statistics and the forward-backward algorithm. The lexical context reflected in decision tree training is more limited since the POS tag or filler indicator features only represent a couple of words adjacent to the word boundary in consideration.

The decision tree's advantage in IP detection can be explained by the pattern match features. Even though the HE-LM also utilizes this information to some extent through

Table 4.11: Boundary event prediction performance on HE-LM and decision tree.

| Transcription | SU | | SU-inc | | IP | | Word-based |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Recall | Precision | Accuracy |
| Reference, HE-LM | 76.48 | 82.38 | 33.84 | 57.51 | 60.93 | 81.72 | 90.78 |
| Reference, DT | 69.84 | 79.31 | 23.48 | 63.64 | 64.18 | 78.46 | 89.93 |
| STT, HE-LM | 72.67 | 76.15 | 30.04 | 37.44 | 51.94 | 64.59 | 88.18 |
| STT, DT | 68.25 | 73.97 | 20.15 | 43.44 | 52.77 | 64.91 | 87.69 |

Table 4.12: Recall rates of IP events for different IP types

| Models | After Edit | | Between Edit&Filler | | Before Filler | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| DT, without Match | 42.90 | 36.84 | 78.06 | 63.78 | 75.65 | 56.81 |
| LM | 43.72 | 42.15 | 76.18 | 63.39 | 74.56 | 57.44 |
| DT, with Match | 48.69 | 44.74 | 78.68 | 64.96 | 76.23 | 56.63 |

trigram counts, it would fare poorly when words unseen in the training data are repeated or when the number of words repeated is long. The effect of the pattern match features becomes more evident when recall rates of different types of IPs are compared. As shown in Table 4.12, the recall rate of IPs following edit regions was actually higher with the HE-LM than with the decision tree trained without the pattern match features. However when the pattern match features were used, the decision tree outperformed the HE-LM.

Examination of detection statistics of IPs occuring before fillers confirmed the trend we observed with decision trees but revealed different detection characteristics. Table 4.13 shows recall rates of those IPs that exist before the five most frequenly occuring fillers. The results obtained with the decision tree trained with prosodic and all lexical features are also included in the table. As was the case with decision trees, IPs before "like" were the hardest to detect. However, the recall rates of IPs before these fillers were much higher with the

Table 4.13: Recall rates for different fillers, on reference transcripts.

| Filler Type | HE-LM | DT |
|---|---|---|
| Filled Pauses | 87.0 | 93.4 |
| You know | 83.5 | 91.3 |
| Like | 31.3 | 16.2 |
| I mean | 85.5 | 82.6 |
| Well | 71.2 | 50.0 |

HE-LM. Moreover the recall rates of IPs before "I mean" and "Well" were higher with the HE-LM, while those before "you know" and filled pauses were lower.

Since the set of filled pauses in our study is limited to five hesitation sounds, the relatively low recall rate of IPs before filled pauses is somewhat surprising. We believe that this result can be explained by our treatment of the case where a filler immediately follows the end of an SU. Since we consider those word boundaries to contain only SU events, it negatively affects the filler detection performance of the HE-LM[5]. The higher recall rate of IPs before "you know" with the decision tree may also be attributed to informative value of prosodic features, but more study is necessary to confirm our hypothesis. On the other hand, the higher recall rates of IPs before "like" and "well" obtained with the HE-LM indicate that longer lexical context is important in detecting these fillers.

Analysis of the IP detection performance around word fragments also supported what we found with decision trees. The recall rates of IPs that followed word fragments were far lower that than of other IPs, and the fact that word fragments were transcribed as full words by the STT system helped the IP recall rates. This particular effect was even more prominent with the HE-LM, as we observed 22.7% recall rate of IPs following word fragments on the STT transcripts versus 17.5% observed on the reference.

[5]The IPs before filled pauses can be inserted in post-processing, as explored in Section 4.6.3.

### *4.5 Model Combination*

As discussed in Section 4.2, we experimented with three different model combination approaches. For the joint tree model, the posterior probabilities of boundary events for each word boundary were used as a feature in decision tree training along with other lexical features and a group of prosodic features selected for this configuration as described in Section 4.3.3. The HE-LM and the decision tree trained with prosodic, POS tag, pattern match and filler indicator features were combined through the independent combination model and integrated HMM approaches. Use of word-based features in decision tree training can violate the independence assumption we made for some of our model combination approaches. However, our experiments showed that strict observation of such assumptions were not necessary, and higher accuracy obtained by using lexical features in decision tree training led to higher performance of combined models. For each of these two combination approaches, we empirically determined the weighting factor $\lambda$ by comparing the percentage of correctly detected boundary events on the development data. Separate weighting factors were found for the reference and STT transcripts. We utilized the SRILM's functionality to perform forward-backward computation for the integrated HMM approach.

The results of our experiments are summarized in Table 4.14 and Table 4.15. The comparable results from the two component models used in combination experiments can be found in the same tables. All three combination approaches achieved improvements in overall accuracy over the individual decision tree model or the HE-LM. However, the combination approaches differed in the way such improvements were gained.

On both reference and the STT transcripts, the SU and IP detection recall rates were the highest with the joint tree model. However, their precision was also the lowest at the same time. The indepedent combination approach resulted in higher precision and lower recall in SU and IP detection than in the joint tree model, but interestingly the opposite case (lower precision and higher recall) was true for the SU-inc event. We believe this result can be explained by the detection performance differences exhibited by the decision tree and the HE-LM, which demonstrate the same trend in prediction recall and precision trade-offs.

The performance differences between the reference and the STT transcripts also illus-

Table 4.14: Model combination experiment results on reference transcripts.

| Model | SU | | SU-inc | | IP | | Word-based |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Recall | Precision | Accuracy |
| Decision Tree | 69.84 | 79.31 | 23.48 | 63.64 | 64.18 | 78.46 | 89.93 |
| HE-LM | 76.48 | 82.38 | 33.84 | 57.51 | 60.93 | 81.72 | 90.78 |
| Joint Tree | 80.86 | 82.69 | 45.12 | 66.37 | 68.17 | 77.94 | 91.68 |
| Independent Combination | 80.44 | 83.62 | 46.65 | 59.77 | 67.91 | 81.99 | 91.98 |
| Integrated HMM | 75.08 | 86.14 | 30.18 | 76.74 | 62.94 | 84.74 | 91.44 |

Table 4.15: Model combination experiment results on STT transcripts.

| Model | SU | | ISU | | IP | | Word-based |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Recall | Precision | Accuracy |
| Decision Tree | 68.25 | 73.97 | 20.15 | 43.44 | 52.77 | 64.91 | 87.69 |
| HE-LM | 72.67 | 76.15 | 30.04 | 37.44 | 51.94 | 64.59 | 88.18 |
| Joint Tree | 77.69 | 76.74 | 39.16 | 41.53 | 57.35 | 61.53 | 88.77 |
| Independent Combination | 75.58 | 76.85 | 40.30 | 37.72 | 56.43 | 65.52 | 88.82 |
| Integrated HMM | 71.32 | 79.43 | 28.52 | 51.37 | 52.16 | 69.45 | 88.90 |

Table 4.16: Recall rates of SU events for different SU types.

| Models | Statement | | Question | | Backchannel | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| Joint Tree | 77.79 | 74.90 | 76.04 | 78.03 | 93.62 | 94.44 |
| Independent Combination | 77.06 | 71.56 | 71.87 | 73.12 | 95.20 | 95.14 |
| Integrated HMM | 69.05 | 64.46 | 67.13 | 67.92 | 93.78 | 92.53 |

trates the weakness of the joint tree model. As the lexical features used in training became much less reliable with word recognition errors in the STT transcripts, its performance degraded, resulting in the lowest overall accuracy of the three combined models with the STT transcripts. Although word recognition errors also affect the other two combination models, the use of the weighting factor allows those models to partially offset the effect of word errors.

The integrated HMM approach achieved the lowest recall rates, but the precision of predictions were the highest with this model. This characteristic resulted in the lowest overall accuracy on the reference transcripts of the three model combinations but led to the highest accuracy on the STT transcripts. The integrated HMM's tendency to increase precision at the cost of recall becomes more evident when we look at individual recall rates of each SU and IP types. As can be seen in Table 4.16, recall rates of statement and question type SU events on the integrated HMM were much lower than those on the other combination models. However, recall rates of backchannel type SU events on the integrated HMM was more in line with the other combination models, perhaps because longer word context is not so important for backchannel detection. Similar observations can be made for IP detection. As shown in Table 4.17, the difference in recall rates between the integrated HMM and other models was the largest for IPs following edit regions, while such differences were less noticeable for other types of IPs.

Table 4.17: Recall rates of IP events for different IP types.

| Models | After Edit | | Between Edit & Filler | | Before Filler | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| Joint Tree | 54.57 | 49.30 | 78.37 | 68.11 | 79.43 | 61.56 |
| Independent Combination | 51.06 | 47.78 | 82.45 | 68.90 | 81.36 | 60.75 |
| Integrated HMM | 44.37 | 42.36 | 79.00 | 64.96 | 77.75 | 57.35 |

We also computed recall rates of IPs that occur before fillers and compared them between different models. The IP recall rate comparisons are shown in Table 4.18. Interestingly, improvements in recall rates over the HE-LM or the decision tree model varied across different combination approaches. For example, although the joint tree model showed increase in recall of IPs before "like" and "well", the recall rates of IPs occuring before other fillers were lower than those achieved with the HE-LM or the decision tree. Similarly, the recall rates of IPs before "like" and "well" were lower with the independent combination model than with the decision tree. Although comparisons of detection performance can not be complete without considering insertion errors, the mixed improvements in recall rates achieved through model combinations suggest that detection strategies may need to be tailored for each conversational fillers for optimal performance.

As expected, the recall rates of IPs after fragments were much lower than those of IPs following other edit regions. Moreoever, compared to improvements seen with other IP types, the recall rates of IPs following fragments showed a relatively small increase through joint tree or independent model combination approaches, which further illustrates the difficulties of detecting this type of IP. The results shown in Table 4.19 also confirm what we observed previously. As demonstrated by the higher recall rate of IPs following fragments on the STT transcripts compared to the reference transcripts, the IP detection improves when word fragments are recognized as full words by the STT system.

Table 4.18: Recall rates for different fillers, on reference transcripts.

| Filler Type | # in Test | DT | HE-LM | Joint Tree | Indep. Comb. | HMM |
|---|---|---|---|---|---|---|
| Filled Pauses | 800 | 93.4 | 87.0 | 89.8 | 96.6 | 95.4 |
| You know | 310 | 91.3 | 83.5 | 87.7 | 91.3 | 90.3 |
| Like | 198 | 16.2 | 31.3 | 47.5 | 22.7 | 14.1 |
| I mean | 62 | 82.6 | 85.5 | 79.7 | 92.8 | 89.9 |
| Well | 52 | 50.0 | 71.2 | 76.9 | 69.2 | 65.4 |

Table 4.19: Recall rates of IPs after fragments vs other edit regions.

| Models | After Fragment | | Other Edits | |
|---|---|---|---|---|
| | Reference | STT | Reference | STT |
| Decision Tree | 25.5 | 23.4 | 67.9 | 56.2 |
| HE-LM | 17.5 | 22.7 | 65.0 | 53.4 |
| Joint Tree | 24.2 | 25.8 | 75.1 | 61.0 |
| Independent Combination | 23.8 | 24.6 | 72.5 | 60.0 |
| Integrated HMM | 18.7 | 19.9 | 66.0 | 54.8 |

## *4.6 Redefining Boundary Events*

### *4.6.1 Merging SU-incomplete into SU category*

As discussed in Section 4.3.4, we found that low detection performance of the SU-inc events were partly due to substitution errors between SU and SU-inc events. Although we initially hypothesized that SU-inc events would be identified with prosodic and lexical characteristics different from other SU types, detecting SU-inc events separately proved to be a problematic task given the highly skewed data and relatively infrequent occurrence of SU-inc events, which represented only about one percent of all boundary events (including null events). Since determining the type of sentences was not among our goals, we experimented merging SU-inc to SU events to see if overall SU detection performance improves.

We trained a decision tree with prosodic and lexical features and an HE-LM to classify each word boundary as one of these three boundary events: SU, IP, and null event. We also combined these two models through the integrated HMM approach. As before, model combination weighting factors were empirically chosen to maximize performance on the development data.

Table 4.20 and Table 4.21 summarize the results of our experiments and compare them to earlier models' on the reference and the STT transcripts. To make performance statistics from earlier experiments comparable to the merged SU event case, SU-inc events were merged to SU events before scoring the predictions. Training the models to predict a single SU event did not yield any significant performance improvement. We noted some changes in SU and IP detection performances, but there was little differences in overall accuracies. In addition, there did not appear to be a noticeable trend except that the precision of SU predictions fell while the precision of IP predictions rose slightly.

To determine whether there are advantages of merging the SU-inc class to the SU, we compared the recall rates of each SU types as shown in Table 4.22. In order to negate the effect of subsitution errors and analyze SU detection performance more objectively, we considered all substitution errors between SU and SU-inc events observed with earlier models to be correct predictions. In other words, a prediction of an SU-inc or an SU event was considered to be correct as long as the true boundary event was an SU or an SU-inc.

Table 4.20: Merging SU-incomplete into SU, on reference transcripts.

| Models | SU | | IP | | Word-based |
|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Accuracy |
| HE-LM | 75.68 | 83.30 | 60.93 | 81.72 | 91.04 |
| HE-LM, merged | 75.49 | 82.99 | 60.67 | 81.73 | 90.96 |
| DT | 70.23 | 82.69 | 64.18 | 78.46 | 90.43 |
| DT, merged | 70.69 | 82.42 | 63.67 | 78.54 | 90.42 |
| HMM | 74.36 | 88.14 | 62.94 | 84.74 | 91.75 |
| HMM, merged | 75.81 | 86.95 | 61.95 | 85.95 | 91.80 |

Table 4.21: Merging SU-incomplete into SU, on STT transcripts.

| Models | SU | | IP | | Word-based |
|---|---|---|---|---|---|
| | Recall | Precision | Recall | Precision | Accuracy |
| HE-LM | 73.25 | 77.39 | 51.94 | 64.59 | 88.60 |
| HE-LM, merged | 73.02 | 76.97 | 51.72 | 64.53 | 88.51 |
| DT | 69.02 | 76.76 | 52.77 | 64.91 | 88.18 |
| DT, merged | 68.98 | 76.93 | 52.81 | 64.66 | 88.21 |
| HMM | 71.66 | 81.40 | 52.16 | 69.45 | 89.29 |
| HMM, merged | 72.22 | 80.58 | 50.94 | 69.69 | 89.26 |

Table 4.22: Recall rates of SU events for different SU types.

| Models | Incomplete | | Statement | | Question | | Backchannel | |
|---|---|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT | Reference | STT |
| DT | 71.34 | 71.86 | 64.01 | 61.87 | 69.36 | 73.12 | 87.26 | 87.76 |
| DT, merged | 72.56 | 76.43 | 64.35 | 61.48 | 70.19 | 73.41 | 87.33 | 87.67 |
| HE-LM | 54.27 | 60.84 | 64.01 | 67.49 | 69.36 | 70.52 | 87.26 | 93.40 |
| HE-LM, merged | 54.88 | 59.69 | 72.07 | 67.25 | 68.52 | 70.23 | 91.90 | 93.49 |
| HMM | 30.18 | 28.52 | 69.05 | 64.46 | 67.13 | 67.92 | 93.78 | 92.53 |
| HMM, merged | 62.20 | 68.82 | 70.94 | 64.96 | 68.52 | 70.52 | 94.45 | 94.44 |

The result showed that SU recall rates generally improved when SU-inc events were merged into the SU event class, but only consistently for the integrated HMM predictor.

Although we did not observe conclusive benefit of merging the SU-inc class to SU events, we will consider all SU types as a single class in our investigation of boundary events in later sections, since it will simplify the analyses of our experiments with classifiers trained to predict different groups of boundary events.

### *4.6.2 Integrated versus individual detection of boundary events*

In our study, thus far we considered SUs and IPs to be mutually exclusive boundary events and train classifiers to distinguish between them, since we believe these two events share some of the prosodic cues such as pauses and pre-boundary vowel lengthening but have different lexical characteristics. Earlier work on boundary detection however focused on either SU or IP detection and made binary decisions between boundary and null event (Shriberg et al. 2000, Liu et al. 2003). As a comparative study, we trained a decision tree, HE-LM and integrated HMM to detect only SU or IP events.

The prediction results of SU-only detection on the reference and STT transcripts are illustrated in Table 4.23. Since making comparison on precision and recall basis is somewhat difficult, the slot error rates (SER), or the sum of insertion and deletion errors divided

Table 4.23: Joint SU-IP vs. SU-only models: SU detection performances.

| Models | Reference | | | STT | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| DT | 70.69 | 82.42 | 44.39 | 68.98 | 76.93 | 51.71 |
| DT, SU only | 69.75 | 83.54 | 44.00 | 68.38 | 77.79 | 51.15 |
| HE-LM | 75.49 | 82.99 | 39.99 | 73.02 | 76.97 | 48.83 |
| HE-LM, SU only | 74.15 | 83.97 | 40.01 | 72.30 | 77.58 | 48.62 |
| HMM | 75.81 | 86.95 | 35.57 | 72.22 | 80.58 | 45.18 |
| HMM, SU only | 73.44 | 88.22 | 36.37 | 69.74 | 82.06 | 45.51 |

by number of true events, of SU events are also shown in the table. The SU detection performances of classifiers discussed in Section 4.6.1 are included as well. Compared to the three-class classifiers, the binary classifiers that only detected SUs generally had lower recall but higher precision. In terms of overall errors, this led to somewhat mixed results. With the HE-LM, the SU-only classifier had slightly more errors on the reference transcripts but less on the STT transcripts. With the decision tree model, the SU-only detection resulted in lower SERs on both types of transcripts. However, with the integrated HMM, the SU-only detection had more errors on both transcripts. Although we may see different results with other combination models, what we observed with the integrated HMM supports joint detection of SUs and IPs.

Similarly, IP detection performances of different models trained to detect only the IP events are summarized and compared in Table 4.24. As discussed in Section 4.1, end of sentences or SUs are often followed by conversational fillers, in which case the word boundary contains both SU and IP events. We decided to treat these boundaries as containing just SU events when we trained our classifiers to predict SU and IP events at the same time. However, those boundaries were considered to have IP events when the prediction target was just the IP. In order to make the IP detection performance of the classifiers trained to predict both SU and IP events more comparable to the IP-only classifiers, the IP predictions

Table 4.24: Joint SU-IP vs. IP-only models: IP detection performances.

| Models | Reference | | | STT | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| DT | 66.76 | 80.74 | 49.16 | 54.63 | 65.88 | 72.28 |
| DT, IP only | 68.04 | 91.40 | 38.36 | 54.23 | 72.45 | 66.39 |
| HE-LM | 64.01 | 83.76 | 48.39 | 53.62 | 65.79 | 72.89 |
| HE-LM, IP only | 65.65 | 89.18 | 42.32 | 55.02 | 69.98 | 68.58 |
| HMM | 65.18 | 87.56 | 44.56 | 52.88 | 70.59 | 67.76 |
| HMM, IP only | 66.39 | 94.07 | 37.79 | 53.70 | 74.79 | 64.40 |

from SU/IP classifiers were postprocessed to include IPs that occur at SU boundaries and before filled pauses, since detecting filled pauses is a trivial task in our work. The results shown in Table 4.24 and Table 4.25 reflect this postprocessing.

Unlike what we observed with SU-only detection, both recall and precision of IP predictions increased with IP-only detection, resulting in significant reduction in SERs. However, a different picture emerges when we look at recall rates of each IP type. As shown in Table 4.25, the recall rates of IPs occuring after edit regions fall while the recall rates of the IPs before fillers increase. This phenomenon can be explained by co-occurrence of SU and IP events, many of which are not captured by the filled pause post-processing rule.

The negative impact on detection of IP events before fillers caused by treating the word boundaries with both SU and IP events as just having SUs becomes even more apparent when recall rates of IPs are examined for different fillers. Table 4.26 shows recall rates of IPs before the five most frequently occurring fillers (and the number of such IPs) in the test data with joint SU-IP vs. IP-only detection setup. Although direct comparisons to results obtained in previous sections (Tables 4.13 and 4.18) are not possible because word boundaries with both SU and IP events had been considered to contain only the SU in the earlier experiments, recall rates of IPs occuring before fillers show clear improvements. The magnitude of improvement is more significant for those fillers that often follow SUs, and

Table 4.25: Joint SU-IP vs. IP only models: Recall rates of IP events for different IP types.

| Models | After Edit | | Between Edit & Filler | | Before Filler | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| DT | 48.69 | 44.74 | 78.68 | 64.96 | 76.23 | 56.63 |
| DT, IP only | 43.60 | 39.98 | 86.56 | 67.32 | 80.41 | 59.62 |
| HE-LM | 43.72 | 42.15 | 76.18 | 63.39 | 74.56 | 57.44 |
| HE-LM, IP only | 42.22 | 40.84 | 85.00 | 70.08 | 77.24 | 60.11 |
| HMM | 43.80 | 41.06 | 79.62 | 64.57 | 75.90 | 56.00 |
| HMM, IP only | 39.30 | 38.79 | 86.56 | 68.90 | 80.15 | 59.13 |

recall rates of IPs before fillers like "I mean" and "well" are substantially better. The IPs before "like" were least affected by explicit modeling of concurrent occurrences of SUs and IPs, and the classifiers fail to recall many of them.

### *4.6.3 Detecting concurrent occurences of SU and IP events*

As discussed in the previous section, our strategy of treating the word boundaries with both SU and IP events as containing just SU events turned out to hurt IP detection performance. Although the issue of SU and IP co-occurence can be resolved by detecting the two events separately, results of our previous experiments suggest that SU detection and prediction of IPs occuring after edit regions improve under the integrated detection framework. In order to increase IP detection performance while preserving the joint SU/IP detection paradigm, we decided to consider the concurrent occurrence of SU and IP events as a new category separate from SU or IP events.

With this new definition of boundary events, we trained decision tree, HE-LM and integrated HMM models to perform classification between SU, SUIP, IP and null events. The detection performances of classifiers trained with these events are shown in Table 4.27 and Table 4.28. Although not presented here, the recall rate of SUIP events was low. However, this was due mostly to the subsitution errors where SUIP events were classified as SU or

Table 4.26: Counts and recall rates for IPs associated with different fillers, on reference transcripts.

| Filler Type | # in Test Data | % of SU/IP co-occurrence | Recall Rates | | |
|---|---|---|---|---|---|
| | | | DT | HE-LM | HMM |
| Filled Pauses | 1054 | 24.1 | 99.6 | 90.1 | 98.5 |
| You know | 450 | 31.1 | 100 | 99.6 | 100 |
| Like | 234 | 15.4 | 8.6 | 30.8 | 17.1 |
| I mean | 139 | 50.4 | 100 | 99.3 | 99.3 |
| Well | 144 | 63.9 | 100 | 80.6 | 95.8 |

IP events. Since we are only interested in detecting SU and IP events and distinguishing SUIP events from SU or IP events was not our goal, predictions made for the SUIP class were merged to either SU or IP predictions when calculating recall, precision and SER of SU or IP events shown in Table 4.27 and Table 4.28. For example, we used the following equation to compute the recall of the SU event:

$$Recall_{SU} = \frac{SU_{SU} + SUIP_{SUIP} + SUIP_{SU} + SU_{SUIP}}{SU_{all} + SUIP_{all}}$$

IP recall, precision and SER numbers presented for the case without the SUIP category come from models trained to predict SU, IP and null events (Section 4.6.1) with filled pause post-processing. Effects of the SUIP category in SU detection were somewhat mixed as its inclusion led to increases in SERs for the decision tree and HE-LM on reference transcription. The IP error rates however improved for all models on both the reference and STT transcripts. Table 4.29 shows IP detection performances from the IP-only predictors and models trained with the SUIP category. Compared to the performances of the IP-only predictors, IP prediction precision was lower with SUIP category except for the HE-LM classifier. Overall error rates however improved for all models due to higher recall rates.

Table 4.27: SU and IP detection performances: IP post-processing vs. with the SUIP category, on reference transcripts.

| Models | SU | | | IP | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| DT | 70.69 | 82.42 | 44.39 | 66.76 | 80.74 | 49.16 |
| DT, w/ SUIP | 70.30 | 82.59 | 44.52 | 70.69 | 88.90 | 38.13 |
| HE-LM | 75.49 | 82.99 | 39.99 | 64.01 | 83.76 | 48.39 |
| HE-LM, w/ SUIP | 75.34 | 82.70 | 40.42 | 66.51 | 89.23 | 41.52 |
| HMM | 75.81 | 86.95 | 35.57 | 65.18 | 87.56 | 44.56 |
| HMM, w/ SUIP | 74.95 | 87.88 | 35.39 | 68.21 | 93.60 | 36.45 |

Table 4.28: SU and IP detection performances: IP post-processing vs. with the SUIP category, on STT transcripts.

| Models | SU | | | IP | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| DT | 68.98 | 76.93 | 51.71 | 54.63 | 65.88 | 72.28 |
| DT, w/ SUIP | 69.15 | 77.17 | 51.30 | 56.58 | 70.83 | 66.72 |
| HE-LM | 73.02 | 76.97 | 48.83 | 53.62 | 65.79 | 72.89 |
| HE-LM, w/ SUIP | 73.00 | 76.66 | 49.22 | 55.32 | 70.78 | 67.52 |
| HMM | 72.22 | 80.58 | 45.18 | 52.88 | 70.59 | 67.76 |
| HMM, w/ SUIP | 72.28 | 80.80 | 44.89 | 54.99 | 74.10 | 64.24 |

Table 4.29: SU/IP/SUIP vs. IP-only: IP detection performance

| Models | Reference | | | STT | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| DT, IP only | 68.04 | 91.40 | 38.36 | 54.23 | 72.45 | 66.39 |
| DT, w/ SUIP | 70.69 | 88.90 | 38.13 | 56.58 | 70.83 | 66.72 |
| HE-LM, IP only | 65.65 | 89.18 | 42.32 | 55.02 | 69.98 | 68.58 |
| HE-LM, w/ SUIP | 66.51 | 89.23 | 41.52 | 55.32 | 70.78 | 67.52 |
| HMM, IP only | 66.39 | 94.07 | 37.79 | 53.70 | 74.79 | 64.40 |
| HMM, w/ SUIP | 68.21 | 93.60 | 36.45 | 54.99 | 74.10 | 64.24 |

Improvements in IP recall rates in comparison to the IP-only predictors are particularly encouraging since the classifiers trained for IP-only detection already showed significant improvement in IP recall rates over other classifiers. When we examine the recall rates of different IP types, we can see that including the SUIP category increases recall rates of all types of IPs[6]. As illustrated in Table 4.30, recall rate increase achieved with the SUIP category over the IP-only detector was relatively greater for IPs following edit regions than IPs before fillers. Interestingly, the integrated HMM trained with the SUIP category also showed improvement in recall rates of IPs occuring after word fragments. Analysis of recall rates of IPs before specific filler words showed little difference between the integrated HMM model trained for IP-only detection and the model trained with the SUIP class, although the decrease in the recall rate of IPs before "like" was more notable.

### *4.6.4 Classifying different IP types*

Although identifying the type of SUs was not one of our goals, distinguishing between different types of IPs - or more specifically between IPs after edit regions and before fillers - could be important in achieving our overall goal of automatic disfluency detection. This

[6]Note that both SUIP and IP predictions were considered to be correct IP detection when calculating IP recall rates for classifiers trained with the SUIP category.

Table 4.30: Recall rates of IP events for different IP types with integrated HMM.

| IP type | HMM | | HMM, IP only | | HMM, w/ SUIP | |
|---|---|---|---|---|---|---|
| | Reference | STT | Reference | STT | Reference | STT |
| After Edit | 43.80 | 41.06 | 39.30 | 38.79 | 43.76 | 41.49 |
| Between Edit& Filler | 79.62 | 64.57 | 86.56 | 68.90 | 87.19 | 70.08 |
| Before Filler | 75.90 | 56.00 | 80.15 | 59.13 | 80.51 | 59.73 |
| After Fragments | 18.11 | 19.53 | 18.53 | 19.14 | 21.89 | 21.48 |

Table 4.31: Recall rates for different fillers, on STT transcripts.

| Filler Type | HMM, IP only | HMM, w/ SUIP |
|---|---|---|
| Filled Pauses | 98.5 | 97.5 |
| You know | 100 | 100 |
| Like | 17.1 | 14.5 |
| I mean | 99.3 | 100 |
| Well | 95.8 | 94.4 |

is because identifying edit regions and conversational fillers are related but separate tasks. Even though detecting IP events without predicting types of the IP events alone will help us locate edit disfluencies and fillers, IP boundary events are likely to be far more valuable with information about its types.

We attempted to predict the types of IP events by training our classifiers to make decisions among these six target events: SU, SUIP, IP-edit, IP-editfiller, IP-filler, and null events. Table 4.32 shows the six-class confusion matrix for the integrated HMM on the reference transcripts. As can be seen in the confusion matrix, a high degree of confusability exists for IP-editfiller and IP-filler events or SUIP events and IP-filler events. However, all these three events are essentially equivalent when they are used to indicate existences of conversational fillers. A similar statement can be made about IP-edit and IP-editfiller events as they both represent ends of edit regions. Therefore, we looked at precision and recall statistics of the following grouped events:

- SU = {SU, SUIP}
- IP-edit = {IP-edit, IP-editfiller}
- IP-filler = {IP-filler, IP-editfiller, SUIP}

Table 4.33 and Table 4.34 show the recall, precision, and SER computed for SU, IP-edit and IP-filler events detected by different models. As was done in the previous section, SUIP events were treated as SUs when calculating error rates for SU events. Similarly, SUIP, IP-editfiller and IP-filler events were all considered to be equivalent to IP-filler events when computing numbers for IP-filler, and IP-edit and IP-editfiller events were calculated together for the IP-edit event.

Compared to what we observed in previous experiments, SU detection performance remains largely the same and results are somewhat mixed. While the SU event SER on the integrated HMM decreased on the reference transcripts, it was higher than what was achieved previously on the STT transcripts. SU detection performance degraded on the decision tree model but improved on the HE-LM. It is difficult to compare IP detection

Table 4.32: Confusion matrix for the integrated HMM, on reference transcripts.

| True events | Detected Events | | | | | |
|---|---|---|---|---|---|---|
| | N | SU | SUIP | IP-edit | IP-editfiller | IP-filler |
| N | 26662 | 1129 | 46 | 592 | 41 | 184 |
| SU | 377 | 3641 | 155 | 96 | 1 | 4 |
| SUIP | 13 | 21 | 427 | 2 | 28 | 76 |
| IP-edit | 57 | 34 | 0 | 529 | 1 | 1 |
| IP-editfiller | 6 | 0 | 18 | 3 | 113 | 48 |
| IP-filler | 34 | 6 | 132 | 4 | 135 | 878 |

Table 4.33: SU, IP-edit and IP-filler detection performances on reference transcripts.

| Models | SU | | | IP-edit | | | IP-filler | | |
|---|---|---|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER | Recall | Precision | SER |
| DT | 71.19 | 81.29 | 45.20 | 44.27 | 70.66 | 74.11 | 82.87 | 94.47 | 21.98 |
| HE-LM | 75.33 | 83.56 | 39.49 | 40.91 | 69.99 | 76.63 | 78.28 | 92.75 | 27.84 |
| HMM | 75.66 | 87.67 | 34.98 | 41.81 | 79.75 | 68.80 | 81.08 | 95.42 | 22.81 |

Table 4.34: SU, IP-edit and IP-filler detection performances on STT transcripts.

| Models | SU | | | IP-edit | | | IP-filler | | |
|---|---|---|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER | Recall | Precision | SER |
| DT | 69.31 | 76.08 | 52.49 | 39.93 | 57.60 | 89.46 | 61.27 | 73.57 | 60.74 |
| HE-LM | 72.88 | 76.96 | 48.93 | 37.72 | 53.62 | 94.90 | 60.41 | 74.11 | 60.70 |
| HMM | 72.51 | 80.44 | 45.12 | 37.72 | 62.71 | 84.71 | 59.79 | 75.79 | 59.31 |

performances to earlier results since it was not possible to calculate precisions or SERs for each IP types separately. However, comparisons of recall rates of different IP types revealed that the recall rate of IPs following edits decreased while the recall rate of IPs before fillers changed little.

### *4.7 Summary*

In this section, we experimented with various information sources and statistical classifiers to detect word boundary events. Experiments conducted in this section largely fell into one of two categories: those performed to evaluate different classifier performances, and those designed to determine an optimum set of classification targets. The classifiers considered in our work included the CART-style decision tree, the HE-LM, and different ways to combine the models of the decision tree and the HE-LM. The classification targets studied included the following:

- SU, SU-inc, IP
- SU, IP
- SU only
- IP only
- SU, SUIP, IP
- SU, SUIP, IP-edit, IP-editfiller, IP-filler

Since evaluating performances of different classifiers under various classification target setups would made managing overall experimental dimension problematic, we chose to evaluate classifier performances under the same classification target setup (namely, SU, SU-inc, IP) before experimenting with different classification targets.

Our study with decision trees showed that successful detection of boundary events using only prosodic features is difficult. The prosodic features we generated were only mildly useful

when detecting boundary events. Although we were able to recall some SU events and IPs that occur before fillers using only the prosodic featuers, we did not achieve reasonable detection performance until lexical features were used in prediction. This finding echoes what was reported by Liu et al. (2003). In their work, a decision tree trained with only the prosodic features was used to detect IPs. Even though they achieved some success with the data downsampled to have equal priors of IP and null events, the prosodic decision tree failed to detect any IP events when tested on actual data with highly skewed distribution of null and IP events. Although our study does suggest that prosodic information is useful in boundary event detection (especially when lexical features become less reliable due to word recognition errors), we need to devise other ways of capturing and representing prosody to take better advantage of this information source.

Among different lexical features used in decision tree training, the POS tag features proved to be most useful. The fact that inclusion of POS tag features improved detection of all boundary event types suggest that syntactic information is very closely related to and important for boundary event detection.

The HE-LM, which utilized word identities and statistically generated contexts to recognize boundary events, was quite successful in detecting SUs and IPs that occur before fillers. However, the decision tree had an edge in detecting IPs following edit regions largely because it could utilize pattern match features. This result illustrates the weakness and strengths of each model. The HE-LM is quite adept at dealing with an information source that has a large number of possible values, but incorporating new information sources in the model is not trivial. The decision tree on the other hand offers a convenient and powerful way to combine different information sources but is limited in its ability to handle a large number of discrete values.

Model combination methods allow us to utilize strengths of different models while compensating for their individual weaknesses. We experimented with three combination approaches. We observed improvements in overall boundary event detection accuracy for all combination approaches, but each approach achieved such improvements in different ways. Performance improvements with the joint tree model were obtained largely through increases in event detection recall rates. The integrated HMM approach on the other hand

favored precision to recall. The independent combination model did not show a strong bias toward recall or precision, but its performance suffered relatively more than other combination models when a component model was a particularly bad predictor for a certain event. This characteristic can be attributed to the single mixture weight used in our combination experiments. We believe its performance would have been higher if we had instead used some function of lexical and prosodic features as the mixture weight.

In that regard, the joint tree model may be thought as a more advanced combination approach since it can model interactions between component models. However, since we only had the reference transcripts of the training data, we were not able to retrain the joint tree model for the STT transcripts. We could however optimize the performances of other combination models on the STT transcripts by tuning their weighting factors. Consequently, the prediction accuracy of the joint tree model was lower than the other two models on the STT transcripts. This result is similar to what was seen in the work of Stolcke et al. (1998). However, compared to their work, performance degradation due to word recognition errors was far less severe for all models since the word error rate on our STT test data was approximately 23% as opposed to over 46% reported in (Stolcke et al. 1998).

We also experimented with detecting different classes of boundary events. We reconsidered our initial decision to treat incomplete SUs as a separate class and experimented with merging them to a single SU category. Although experimental results of merging SU classes were inconclusive, we chose to use a single class to represent all SU types, since doing so simplified analyses of subsequent boundary event experiments. In order to verify the benefits of detecting SU and IP events with a single classifier, we trained classifiers to detect the SU events and the IP events independently. The experimental results showed that joint prediction of SU and IP events improves detection of SU events and IPs that occur after edit regions. However, we discovered that the detection of IPs before fillers was negatively impacted by ignoring IPs when SU and IP events occur at the same time. Our later experiments demonstrated that the issue of concurrent occurence of SUs and IPs can be resolved by detecting them in a new SU-IP class separate from isolated SU and IP events. In addition, we trained classifiers to distinguish between different IP types. Although such a

Table 4.35: Boundary event detection performance scored with standard scoring tools.

| Condition | SU | | | IP | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| reference, rteval | 75.68 | 88.11 | 34.53 | 66.75 | 92.87 | 38.38 |
| reference, su/df-eval | 75.86 | 88.45 | 34.04 | 67.90 | 93.83 | 36.56 |
| STT, rteval | 67.45 | 81.42 | 47.95 | 48.55 | 77.83 | 65.28 |
| STT, su/df-eval | 67.45 | 83.37 | 45.97 | 46.31 | 74.06 | 69.91 |

prediction strategy does not yield improvements in boundary event detection performance, we believe that IP predictions can be more useful when identifying fillers and edit regions if the information about types of IP events are available.

Table 4.35 summarizes the boundary event detection performance of the best-case system which uses the integrated HMM system trained to predict six classes including SU, SUIP, IP-edit, IP-editfiller, IP-filler, and Null. The IP detection performance were scored with rteval-v2.3 and df-eval-v13, and the SU detection performance was scored with rteval-v2.3 and su-eval-v15. Except the SUIP class, which was treated as both SU and IP-filler events, no other postprocessing was done to the boundary event predictions. Note that the results shown in Table 4.35 are slightly different from numbers presented previously, which were not based on these tools, due to differences in aligning reference boundary events to system output (which also partly explain the differences between the tools).

# Chapter 5

# IDENTIFYING EDIT REGIONS AND CONVERSATIONAL FILLERS

After boundary events have been detected, we use rules learned through transformation-based learning (TBL) to detect edit regions and conversational fillers. TBL is an automatic rule learning algorithm that has been used for a variety of problems in natural language processing. We will describe TBL in more detail in Section 5.1 and introduce our TBL system setup in Section 5.2. Experimental paradigm and results are discussed in Section 5.3. In Section 5.4, we will summarize results and discuss directions for future work. The work presented in this chapter reflects collaborative work with Sarah Schwarm.

## *5.1 Transformation-Based Learning*

TBL is an iterative machine learning technique for inducing rules from data. TBL is an error-driven learning algorithm designed for supervised learning, and it has been successfully used in many corpus-based natural language processing and linguistic modelling tasks such as part-of-speech tagging (Brill 1995), spelling correction (Mangu & Brill 1997), error correction in automatic speech recognition (Mangu & Padmanabhan 2001), and named entity detection (Kim & Woodland 2000).

A TBL system consists of a baseline predictor, a set of rule templates, and an objective function. The baseline predictor is used to make initial predictions for data in the training corpus. The rule templates and objective function are used to greedily search for a best order of rules to apply to the initial predictions. This search is done iteratively as follows. At each iteration of learning, rules are generated accoring to rule templates and used to transform predictions made in the previous iteration. Then the rule whose application results in the best score according to the objective function is added to the ordered list of transformations. The learned rule is applied to the training corpus in preparation for the next iteration. The rule learning stops when the score of the best transformation falls below

a threshold value. Other stopping criteria such as statistical significance measures have also been used (Mangu & Padmanabhan 2001).

There are several possible choices for the baseline predictor, including a rudimentary classifier that uniformly assigns the most likely prediction, random decisions, or a human annotator who assigns initial guesses. In our work, we use a rudimentary classifier. An objective function can be any metric that can evaluate transformations, but the token error rate is commonly used, and we also use this objective function. Every transformation rule in TBL consists of a triggering environment and a rewrite rule. An example of a *learned transformation rule* is shown below:

$$\overbrace{\text{If current word is ``a'' and next word is a possesive pronoun,}}^{\text{Triggering environment}} \Rightarrow \overbrace{\text{change tag to edit region}}^{\text{Rewrite rule}}$$

The rule templates specify which attributes of the data can be used in the triggering enviroment to transform predictions. The *rule template* used to generate the transformation rule in the above example would have the identity of current word and POS tag of the next word as attributes for the triggering environment, as shown below:

$$\overbrace{\text{Word}(0),\ \text{POS}(1)}^{\text{Triggering environment}} \Rightarrow \overbrace{\text{Tag}(0)}^{\text{Rewrite}}$$

Once a list of transformation rules are learned from the training data, classification of new data is performed by applying the rules to the data in the order they were learned. Preserving the order of transformations allows decisions made by earlier rules to be utilized by other rules later in the process. TBL's ability to use intermediate predictions in later transformations to fine-tune classification decisions differentiates it from decision trees, and can be advantageous when there are sequential dependencies in the targets.

### *5.2 Detecting Edit Regions and Fillers with TBL*

In our study, detecting edit regions and conversational fillers with TBL was structured as a tagging problem. Each word in the training data was tagged as one of the five possible values which included "edit region", "filled pause", "discourse marker", "explicit editing term", and "none". The TBL training started by applying the baseline predictor to the data, which

assigned the most common tag value "none" as initial guesses to all words. Transformation rules were then generated and applied to the data. The transformed predictions were compared to true tags, and token error rates were calculated to score the rules. Rule learning continued as described in the previous section until the error reduction count fell below a threshold of 5.

Based on our knowledge of characteristics of edit regions and conversational fillers as defined in our study, we generated following the lexical features for our TBL system:

- Identity of word.
- Part of speech (POS) and grouped POS (GPOS) of the word
- Multi-value flag that indicates whether the current word is commonly used as a filled pause, backchannel, explicit editing term or discourse marker.
- Flags indicating word, POS, and GPOS matches.
- Flags indicating beginning of a turn or an SU.
- Tag to be learned.

The POS and GPOS features are the same as the ones used in word boundary event detection (Section 4.3.2). Use of the flag that signals existence of potential filler and backchannel words was motivated by the fact that annotations in our data were based on closed sets of filled pauses, discourse markers and backchannel words. The flag that indicates word matches was set by comparing the current word to the word up to 3 positions to its right. When there was a match, the relative position of the closest matching word (1, 2, or 3) was assigned to the flag. The POS and GPOS tags were compared and match flags for them were set in the same way. While the match flags were primarily designed to aid edit region detection, flags indicating beginning of a turn or an SU were generated because of speakers' propensity to start utterances with fillers during conversations. The turn-boundary

feature described in Section 4.3.1 was used to identify turn beginnings. The tag-to-be-learned feature is different from other features since its values can be changed during the TBL training or classification process. In fact, it is not strictly a feature that reflects a certain information source but instead a target that we want our system to learn. However, as described in Section 5.1, TBL can utilize intermediate predictions when deciding on or applying transformation rules, and our rule templates made use of this capability of TBL.

Word boundary events are another important source of information in edit region and filler detection. Although it is possible to treat them as word-based features, we incorporated them in our TBL system by inserting special words representing the boundary events in the training data. These special words were tagged as "boundary events", and POS and GPOS tags for them were the same as the words themselves (i.e. POS of an SU event was SU). Under this setup, boundary event informations are essentially reflected in the word identity and POS tag features. For obvious reasons, the special words representing boundary events were ignored when generating word/POS/GPOS match flags.

A set of rule templates were created with the features described above. The rule templates were designed to account for individual features of the current word and its neighbors, the proximity of potential conversational fillers to the current word, and word/POS/GPOS matches, especially near a boundary event or potential filler. A total of 141 different rule templates were used to train the TBL systems.

TBL system outputs were postprocessed to eliminate edit region predictions not associated with IP predictions. Since detecting IPs that occur after edit regions is a particularly difficult task, we hypothesized that edit regions predicted solely by TBL systems that did not have access to prosodic information or contextual cues derived from language models are far more likely to be false alarms. The IP predictions to which edit regions were not assigned by TBL systems were ignored as well, because we suspected that those IP predictions might be insertion errors. We confirm this conjecture in an analysis of post-processing alternatives in Section 5.3.4.

## *5.3 Experiments*

### *5.3.1 Experimental Paradigm*

We used Florian and Ngai's Fast TBL system (fnTBL) to learn transformation rules for edit and filler detection. The corpus used to train our TBL system was the LDC1.3 set.

Based on our study of boundary events discussed in Section 4.6, we decided to consider two different definitions of boundary events. In one definition, boundary events were consisted of an end of a sentence (SU), end of a sentence followed by a filler (SUIP), IP after edit regions or before fillers (IP), or other word boundary (null). In another definition, the IP event was further divided into IPs after edit regions (IP-edit), IPs after an edit and before a filler (IP-editfiller), and IPs before fillers (IP-filler), increasing the total number of boundary events to six. Although experiments in Section 4.6.4 suggested that recall rates of IPs after edit regions may decrease when a separate event is used for each IP type, we hypothesized that separating IP events could increase detection performance of edits and fillers since TBL systems might make a better use of IP events if they had access to IP type information even though such information may be noisy.

Since filler words can exist within an edit region, it is possible for a word to be both a filler and part of an edit region. Our TBL setup does not allow for such a word to be tagged as both a filler word and a part of edit region. For those cases, the word was treated only as a filler. Although this approach is not ideal, we do not believe this is a siginificant problem since only approximately 0.5% of all filler words and words that belongs to edit regions in the LDC1.3 training data are both filler and edit words.

Among various boundary event prediction models we experimented with in Section 4, we used the joint tree (JT) model and integrated HMM to predict boundary events on training and test data. Since we also had access to human-annotated boundary events for our training data, we trained a TBL system with true boundary events as well. Unless otherwise noted, the performances of TBL systems were evaluated with a scoring tool developed for the NIST Rich Transcript evaluation task, namely rteval v2.3.

### *5.3.2 Prediction Results*

Edit region and filler detection performances of TBL systems trained with true boundary events are summarized in Table 5.1 and Table 5.2, and performances of TBL systems trained with predicted boundary events are shown in Table 5.3 and 5.4. The slot error rates (SER) from the different TBL systems on reference and STT transcripts are further illustrated in Figure 5.1 and Figure 5.2. Note that "1 IP" and "3 IP" in the prediction model column of the tables represent 4-class (with one IP class) and 6-class boundary event classifications respectively.

Although conversational filler detection performance was promising, our TBL systems had far less success in identifying edit regions. Compared to the SERs observed for IP predictions in Section 4.6.4, filler detection performance generally showed improvement, while the SERs for edit region detection were lower than those seen for IPs after edit regions. This result is not surprising since identifying words in edit regions is an inherently more difficult task. Moreover, the use of a closed set for filler words in our study implies that TBL systems have better chance of improving upon the imperfect predictions of IP events before fillers.

The effect of incorrect IP event predictions on edit region and filler detection by TBL systems can be observed in performance differences between the joint tree model and integrated HMM as well as between TBL systems trained with true and predicted boundary events. With TBL systems trained with true boundary events, boundary events predicted with the integrated HMM result in lower SERs on both reference and STT transcripts even though higher overall prediction accuracy was observed with the joint tree model than with the integrated HMM on reference transcripts in our comparisons of different boundary event prediction models in Section 4.5. The recall and precision statistics shown in Table 5.1 and Table 5.2 mirror the same trend we have seen in boundary event detection. The recall rates were higher with joint tree model, but the integrated HMM had better precision. However, although the joint tree model's prediction bias led to higher overall boundary event detection accuracy on reference transcripts, lower precision boundary event predictions made by the joint tree model hurt edit and filler detection performance more severely as the TBL

systems trained with true boundary events learn to rely on boundary events.

We observe slightly different results when the TBL systems were trained with predicted boundary events. As can been seen in Table 5.3, the SER of edit and filler detection on the reference transcripts with the "1 IP" condition was lower with the joint tree model than with the integrated HMM. Since the precision and recall trend remains the same, this result indicates that TBL systems can learn to negate the effects of IP prediction errors. Using the boundary events predicted with the integrated HMM still resulted in lower edit region SERs on the STT transcripts as well as lower filler SERs on both transcripts. In all cases, performances improved when TBL systems were trained with predicted boundary events.

Filler detection accuracy increased when IP events were separated into three different types. However, predicting different types of IP events had mixed results on edit region detection. When the integrated HMM was used to predict the boundary events, detecting three different types of IP events resulted in lower recall and higher precision of edit region detection, and overall SERs decreased slightly. However, when the joint tree model was used, edit region recall rates increased and precision decreased, resulting in lower overall detection performance and higher SERs. In spite of the mixed results we observed with the joint tree model, we believe that improvements seen with the integrated HMM for both edit and filler detection support our hypothesis that information about IP event types is useful to TBL systems in edit and filler detection. In Section 5.3.4, we will discuss the impact of the post-processing rules on IP detection performance.

Edit and filler detection performances degraded substantially when the TBL systems were used detect edit regions and fillers in the STT transcripts. The filler detection performance was impacted relatively more by word recognition errors, which is not very surprising since word identity information is very important for the filler detection. As can be seen in Figure 5.1 and Figure 5.2, the joint tree models suffer larger performance loss compared to the integrated HMM models. This result confirms what we observed in Section 4.5. While the integrated HMM prediction performance is adjusted for the imperfect STT transcripts by calibrating the component weighting factor, the joint tree model is trained only with features generated from correct word hypotheses and timings. Consequently the degree of performance degradation caused by word recognition errors in the STT transcripts is more

Table 5.1: Edit and filler detection results on TBL system trained with true boundary events on reference trancripts.

| Boundary Event Prediction Model | Edit Region | | | Filler Words | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | 40.12 | 64.03 | 82.41 | 83.31 | 94.86 | 21.21 |
| JT, 3 IP | 40.43 | 62.11 | 84.23 | 85.43 | 93.98 | 20.04 |
| HMM, 1 IP | 32.62 | 71.10 | 80.63 | 83.27 | 95.28 | 20.85 |
| HMM, 3 IP | 31.04 | 75.54 | 79.01 | 85.14 | 95.04 | 19.30 |

Table 5.2: Edit and filler detection results for TBL systems trained with true boundary events on STT trancripts.

| Boundary Event Prediction Model | Edit Region | | | Filler Words | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | 29.49 | 53.02 | 96.64 | 64.36 | 81.90 | 49.86 |
| JT, 3 IP | 30.42 | 51.20 | 98.57 | 65.67 | 81.12 | 49.61 |
| HMM, 1 IP | 25.05 | 60.45 | 91.34 | 64.11 | 82.52 | 49.47 |
| HMM, 3 IP | 23.62 | 65.91 | 88.60 | 65.17 | 82.27 | 48.87 |

severe for the joint tree model.

### *5.3.3 Analysis of Learned Rules*

Analyzing rules learned by TBL systems are complicated since decisions made early in the process can be used or changed by transformation rules later. However observations can be made about the trends and characteristics of rules chosen for different training configurations. We offer such high-level observations in this section. Since we did not train different TBL systems for the STT transcripts, there is no rule analysis to be done for performance differences between the reference and STT transcripts.

Table 5.3: Edit and filler detection results for TBL systems trained with predicted boundary events on reference trancripts.

| Boundary Event | Edit Region | | | Filler Words | | |
|---|---|---|---|---|---|---|
| Prediction Model | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | 37.61 | 71.70 | 77.23 | 83.56 | 95.52 | 20.36 |
| JT, 3 IP | 38.81 | 66.01 | 81.18 | 85.60 | 94.51 | 19.37 |
| HMM, 1 IP | 31.31 | 77.29 | 77.89 | 83.49 | 96.18 | 19.83 |
| HMM, 3 IP | 30.34 | 79.29 | 77.58 | 84.86 | 95.85 | 18.81 |

Table 5.4: Edit and filler detection results for TBL systems trained with predicted boundary events on STT trancripts.

| Boundary Event | Edit Region | | | Filler Words | | |
|---|---|---|---|---|---|---|
| Prediction Model | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | 27.29 | 60.91 | 90.22 | 64.61 | 82.44 | 49.15 |
| JT, 3 IP | 29.03 | 54.30 | 95.40 | 65.56 | 81.31 | 49.51 |
| HMM, 1 IP | 23.70 | 67.59 | 87.67 | 64.36 | 82.95 | 48.87 |
| HMM, 3 IP | 23.46 | 69.45 | 86.86 | 64.93 | 82.88 | 48.48 |

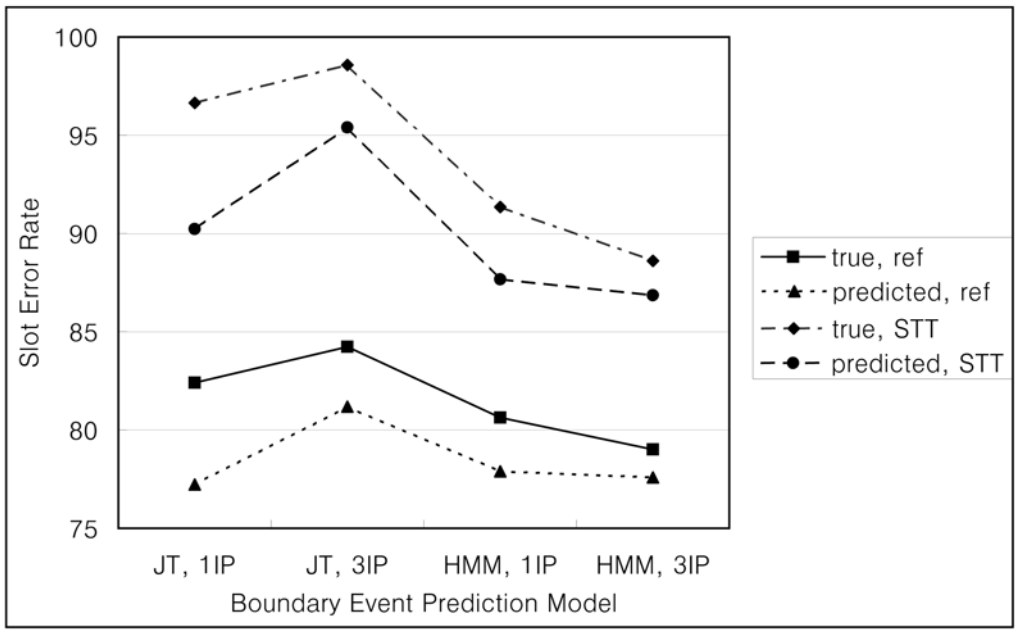


Figure 5.1: Edit region detection slot error rates.

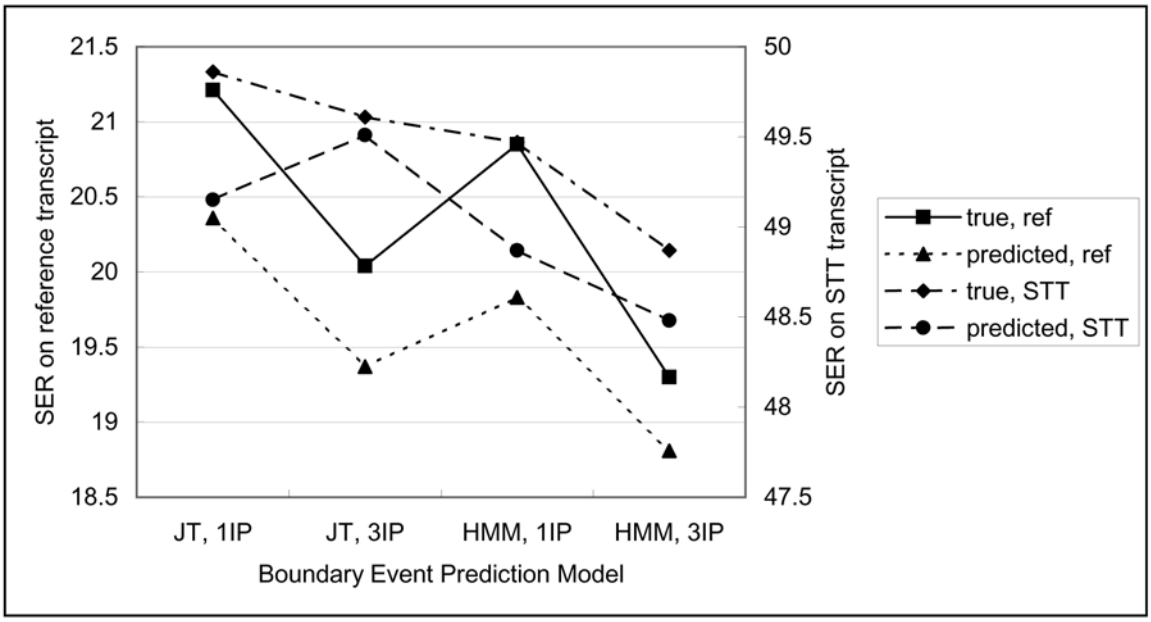


Figure 5.2: Filler detection slot error rates.

Reviewing rules learned with true boundary events, we found that the flag indicating the existence of known field pause words and IP events were used at the top of transformation rule lists. The boundary events and word and POS tag match features were used most frequently early in the process, and rules that were triggered by certain POS and grouped POS tags as well as words generally were found toward the end. When specific POS tags or words were used, they were usually used in conjunction with prior guesses or boundary event information in most cases.

One notable difference between the TBL system trained with a single IP event class and the one trained with three IP event types was the number of transformation rules in the list. The TBL system trained with a single IP class had 212 learned rules whereas the other system had 151 rules. Further examination of the learned rules revealed that the TBL sytem trained with a single IP class guessed every word before an IP event to be part of an edit region early on and applied a number of rules later to correct that initial guess. On the other hand, the TBL system trained with three IP classes used mostly the IP-edit and IP-editfiller cues when guessing edit regions. The way that majority of filler words were detected also differed between the two systems, with the TBL system trained with a single IP class relying on filler indicator flags and word identities in addition to IP events. The other TBL system showed more dependence on filler-related IP events.

The overall trend of rules learned by TBL systems trained with predicted boundary events were similar to those learned by systems trained with true boundary events, especially near the top of transformation rule lists. However, the TBL systems trained with predicted boundary events tended to utilize word identity information and word match feature far more frequently. The number of learned rules were not significantly different from those learned by true-event TBL systems, and training with a single IP class also resulted in longer transformation rule lists.

### *5.3.4 Error Analysis*

Although experimental results presented in the previous section demonstrate edit and filler detection performances of our overall system, it is difficult to assess effectiveness of TBL

Table 5.5: Theoretical edit and filler detection performance with TBL. TBL systems were trained with true boundary events and applied on reference trancripts with true (oracle) vs. predicted boundary events.

| Boundary Event Prediction Model | Edit Region | | | Filler Words | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER |
| oracle, 1 IP | 68.03 | 84.21 | 44.72 | 89.20 | 97.95 | 12.67 |
| HMM, 1 IP | 31.31 | 77.29 | 77.89 | 83.49 | 96.18 | 19.83 |
| oracle, 3 IP | 79.09 | 94.42 | 25.59 | 98.09 | 98.58 | 3.32 |
| HMM, 3 IP | 30.34 | 79.29 | 77.58 | 84.86 | 95.85 | 18.81 |

systems since the boundary event predictions were far from being accurate, especially for IPs after edit regions. In order to gauge the performance of our TBL systems in a more controlled and isolated setting, we applied learned transformation rules to the reference transcripts with true boundary events (oracle case). Table 5.5 compares the oracle edit and filler detection performances to the detection performances seen from the TBL system trained with boundary events predicted by the integrated HMM classifiers. The results reveal that TBL systems can successfully detect a large number of edit regions and the majority of conversational fillers with high precision given accurate boundary event information. Filler predictions were highly accurate, especially when the types of IP events were specified. Although substantial increase in prediction accuracy was observed for edit detection as well, edit detection performance did not reach the levels seen with fillers. The oracle TBL systems were especially ineffective for longer edit regions that exhibited no clear lexical or syntactic matching patterns.

This result suggests that TBL can be used to identify conversational fillers successfully, if the boundary event prediction accuracy was improved. In addition, the oracle experiment also shows the potential for bigger gains with the 3 IP model. Edit region detection also benefits from improvement in boundary event detection accuracy, but identifying longer edit regions such as restart edit disfluencies remain a challenging task.

As discussed in Section 5.2, we postprocessed TBL system outputs to remove edit region

Table 5.6: Effects of post-processing on edit region detection for TBL systems trained with true boundary events.

| Boundary Event Prediction Model | Post-processing | Reference | | | STT | | |
|---|---|---|---|---|---|---|---|
| | | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | Remove Both | 40.12 | 64.03 | 82.41 | 29.49 | 53.02 | 96.64 |
| | Edit Preserved | 40.12 | 63.84 | 82.61 | 29.49 | 52.91 | 96.75 |
| | IP Added | 40.39 | 61.36 | 85.04 | 29.73 | 50.49 | 99.42 |
| JT, 3 IP | Remove Both | 40.43 | 62.11 | 84.23 | 30.42 | 51.20 | 98.57 |
| | Edit Preserved | 40.43 | 61.97 | 84.38 | 30.42 | 51.10 | 98.69 |
| | IP Added | 40.47 | 61.81 | 84.54 | 30.42 | 51.07 | 98.72 |
| HMM, 1 IP | Remove Both | 32.62 | 71.10 | 80.63 | 25.05 | 60.45 | 91.34 |
| | Edit Preserved | 32.62 | 70.86 | 80.79 | 25.05 | 60.39 | 91.38 |
| | IP Added | 32.82 | 69.53 | 81.56 | 25.20 | 58.74 | 92.50 |
| HMM, 3 IP | Remove Both | 31.04 | 75.54 | 79.01 | 23.62 | 65.91 | 88.60 |
| | Edit Preserved | 31.04 | 75.40 | 79.09 | 23.62 | 65.77 | 88.67 |
| | IP Added | 31.04 | 75.54 | 79.01 | 23.62 | 65.91 | 88.60 |

predictions without corresponding IP predictions. Furthermore, the IP predictions for which TBL systems failed to detect any edit regions were ignored. We will refer to the edit region hypothesis with no corresponding IP prediction the Edit-noIP, and the IP prediction without edit region as the IP-noEdit.

In order to verify that our post-processing strategy yielded detection performance improvements, we changed the way TBL system output was processed as follows. In one post-processing setup, the Edit-noIP's in the system output but still igonored the IP-noEdit. In another setup, we kept the Edit-noIP was eliminated from the system output, but we marked every word immediately preceding the IP-noEdit as being in an edit region. Experiments were run for all the different system configurations, with results given in Tables 5.6 and 5.7. The experimental results are also illustrated in Figures 5.3, 5.4, 5.5, and 5.6.

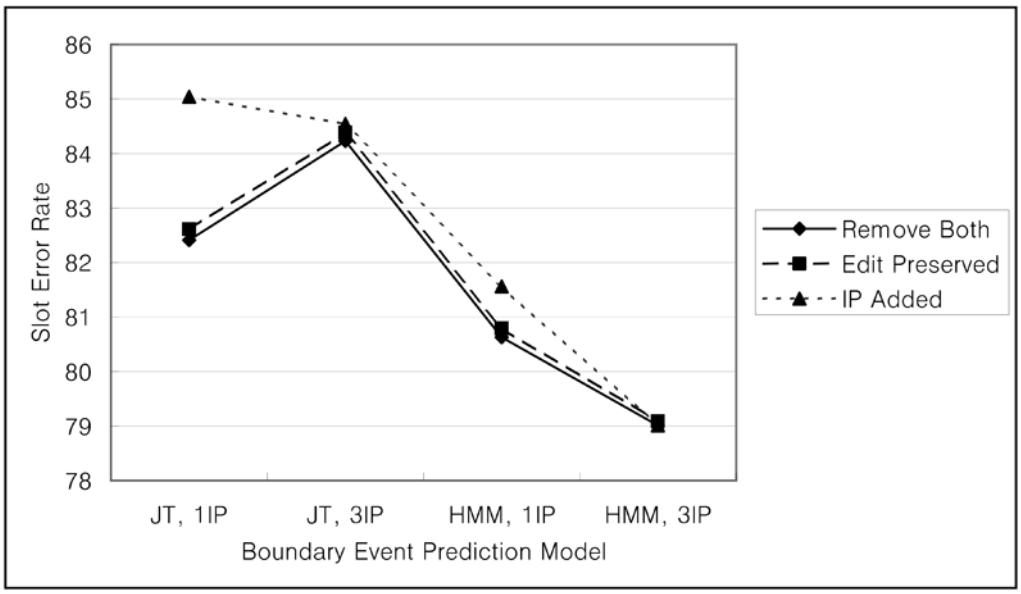


Figure 5.3: Effects of post-processing on edit region detection for TBL systems trained with true boundary events, reference transcripts.

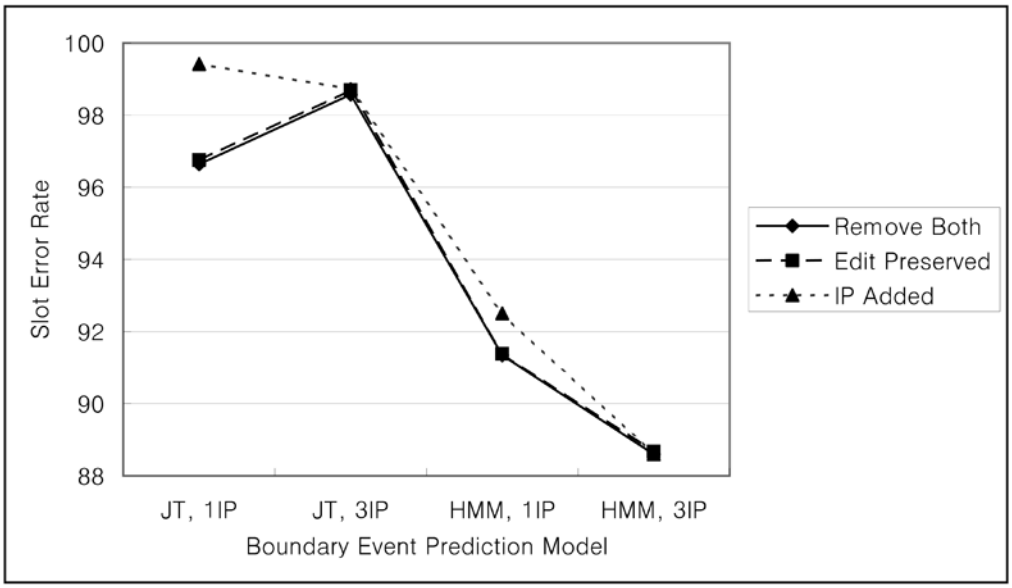


Figure 5.4: Effects of post-processing on edit region detection for TBL systems trained with true boundary events, STT transcripts.

Table 5.7: Effects of post-processing on edit region detection for TBL systems trained with predicted boundary events.

| Boundary Event Prediction Model | Post-Processing | Reference | | | STT | | |
|---|---|---|---|---|---|---|---|
| | | Recall | Precision | SER | Recall | Precision | SER |
| JT, 1 IP | Remove Both | 37.61 | 71.70 | 77.23 | 27.29 | 60.91 | 90.22 |
| | Edit Preserved | 38.07 | 71.17 | 77.35 | 27.64 | 60.85 | 90.14 |
| | IP Added | 39.20 | 63.93 | 82.91 | 28.80 | 53.64 | 96.10 |
| JT, 3 IP | Remove Both | 38.81 | 66.01 | 81.18 | 29.03 | 54.30 | 95.40 |
| | Edit Preserved | 39.16 | 65.82 | 81.18 | 29.11 | 54.09 | 95.59 |
| | IP Added | 38.81 | 65.15 | 81.95 | 29.07 | 53.79 | 95.90 |
| HMM, 1 IP | Remove Both | 31.31 | 77.29 | 77.89 | 23.70 | 67.59 | 87.67 |
| | Edit Preserved | 35.14 | 72.66 | 78.08 | 26.29 | 63.73 | 88.67 |
| | IP Added | 31.62 | 74.73 | 79.24 | 23.97 | 64.58 | 89.18 |
| HMM, 3 IP | Remove Both | 30.34 | 79.29 | 77.58 | 23.46 | 69.45 | 86.86 |
| | Edit Preserved | 33.86 | 74.55 | 77.70 | 25.94 | 65.15 | 87.94 |
| | IP Added | 30.34 | 78.50 | 77.97 | 23.50 | 68.86 | 87.13 |

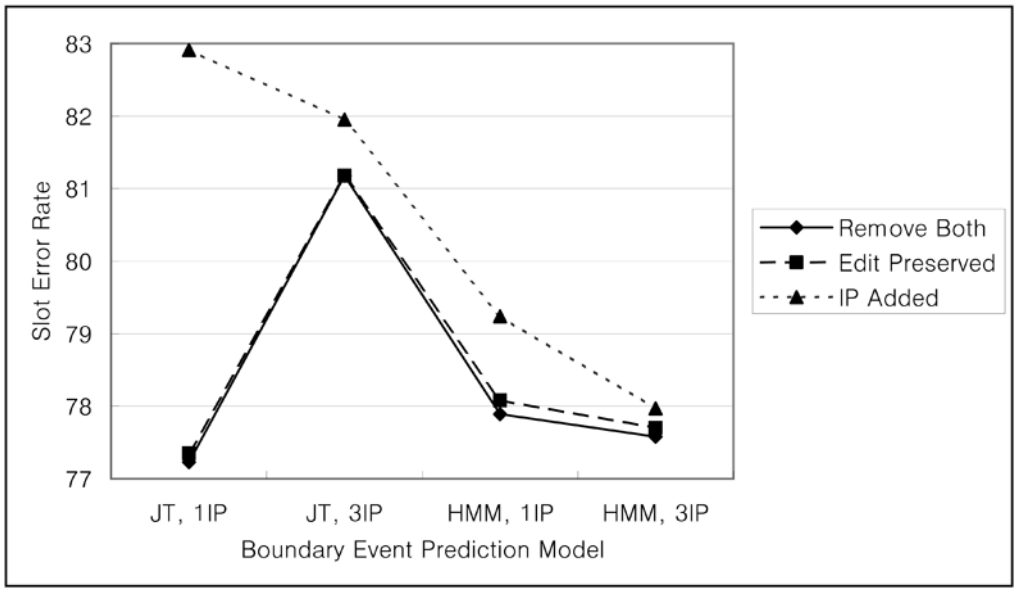


Figure 5.5: Effects of post-processing on edit region detection for TBL systems trained with auto boundary events, reference transcripts.

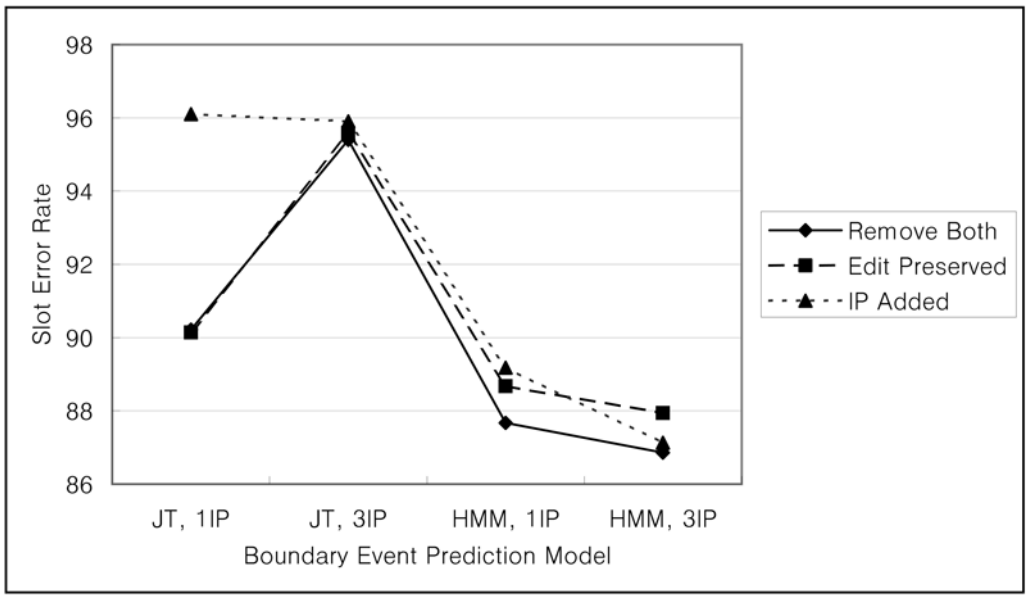


Figure 5.6: Effects of post-processing on edit region detection for TBL systems trained with auto boundary events, STT transcripts.

Overall, the performance differences are small, but the best results are obtained by removing both the Edit-noIP and IP-noEdit cases for all but one system configuration. As can be seen in Table 5.6, preserving the Edit-noIP yielded no improvement in edit region recall rates but degraded detection precision for TBL systems trained with true boundary events. Adding an edit region for the IP-noEdit slightly increased edit region recall rates, but overall SERs increased as well due to lowered precision. Similar trends were observed for TBL systems trained with predicted boundary events. As shown in Table 5.7, inserting an edit region for the IP-noEdit resulted in little or no improvement in edit region recall and ultimately increased SERs because of lowered precisions. The SERs were also higher when the Edit-noIP was preserved, but unlike what we saw with TBL systems trained with true boundary events, keeping the Edit-noIP at least improved recall rates.

Tables 5.8 and 5.9 show comparisons of IP detection performances between TBL systems and the integrated HMM under the 3 IP condition. As expected, detection statistics of IPs after edit regions for the TBL system trained with true boundary events is the same as those for the integrated HMM. Since the TBL system learns to rely on and use every IP events after edit regions when it is trained with true boundary events, this result is not surprising. However, we observe a slightly different result for the TBL system trained with predicted boundary events. This is because the TBL system learns to rely less on IP event information and more on other lexical cues.

For detection of IPs before fillers, TBL systems achieve improvements over the integrated HMM whether they were trained with true boundary events or predicted ones. The TBL system trained with predicted boundary events shows greater improvement in IP detection precision, while the TBL system trained with true boundary events demonstrates higher recall rate. Altogether, the IP post-processing performed after edit and filler detection and improved filler predictions from TBL systems lead to decreases in overall IP SERs in all cases.

We also analyzed recall rates of IPs after fragments since we suspected that TBL systems might have more difficulty identifying edit regions consisting only of word fragments. However, as can be seen in Table 5.10, we did not observe any performance degradation particular to those IPs occuring after fragments. We believe this result may be explained by

Table 5.8: IP detection performance comparisons between TBL systems, using 3 IP HMM in prediction, on reference trancripts.

| Boundary Event Used in TBL | After Edit | | | Before Filler | | | Overall SER |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER | |
| True | 41.81 | 79.75 | 68.80 | 82.69 | 95.27 | 21.42 | 37.28 |
| Predicted | 42.07 | 79.27 | 68.93 | 82.47 | 95.88 | 21.07 | 37.05 |
| no TBL (HMM only) | 41.81 | 79.75 | 68.80 | 81.08 | 95.42 | 22.81 | 38.38 |

Table 5.9: IP detection performance comparisons between TBL systems, using 3 IP HMM in prediction, on STT trancripts.

| Boundary Event Used in TBL | After Edit | | | Before Filler | | | Overall SER |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER | |
| True | 37.72 | 62.71 | 84.71 | 60.55 | 75.56 | 59.03 | 64.74 |
| Predicted | 37.98 | 62.34 | 84.96 | 60.46 | 75.94 | 58.69 | 64.68 |
| no TBL (HMM only) | 37.72 | 62.71 | 84.71 | 59.79 | 75.79 | 59.31 | 65.28 |

the fact that one of the first rules TBL systems learn for edit region detection is to classify every word immediately preceding an IP event predicted to occur after edit regions as part of an edit region. Even though TBL systems apply other transformation rules to change that initial prediction, it appears that such reclassifications do not affect word fragments. However, the recall rates of IPs after fragments were higher on the STT transcripts than on the reference. As discussed in Section 4.3.4, we believe this is because of the fact that the STT system recognizes word fragments as full words when it does recognize them. Since the substitution error rate in recognizing word fragments as full words will not be 100%, lexical features used in our systems are likely to become less noisy with full words than fragments.

Table 5.10: Recall rates of IPs after fragments vs. other edit regions, using 3 IP HMM in prediction.

| Boundary Event Used in TBL | After Fragment | | Other Edits | |
|---|---|---|---|---|
| | Reference | STT | Reference | STT |
| no TBL | 9.9 | 13.7 | 56.0 | 44.4 |
| true | 9.9 | 13.7 | 56.0 | 44.4 |
| auto | 9.9 | 13.7 | 56.4 | 44.7 |

Table 5.11: Edit and filler detection results evaluated by standard scoring tools. The TBL system was trained with boundary events predicted by the integrated HMM, with 3 IP classes.

| Scoring Condition | Edit Region | | | Filler Words | | | IP |
|---|---|---|---|---|---|---|---|
| | Recall | Precision | SER | Recall | Precision | SER | SER |
| rteval, reference | 30.34 | 79.29 | 77.58 | 84.86 | 95.85 | 18.81 | 37.05 |
| df-eval, reference | 30.54 | 79.88 | 77.16 | 85.40 | 96.52 | 17.68 | 35.55 |
| rteval, STT | 23.46 | 69.45 | 86.86 | 64.93 | 82.88 | 48.48 | 64.68 |
| df-eval, STT | 23.35 | 69.11 | 87.09 | 65.17 | 83.89 | 47.35 | 65.21 |

### *5.4 Summary*

We trained TBL systems use boundary event predictions and other lexical information such as word and POS tag identities and pattern match cues to identify edit regions and conversational fillers. Our experimental results showed that identifying edit regions is a particularly difficult task. Relatively high accuracy was achieved for conversational filler detection, but detection performance degraded severely on the STT transcripts. Table 5.11 shows edit and filler detection performance of the best case TBL system which was trained with boundary events predicted by the integrated HMM classifier. Standard scoring tools rteval-v2.3 and df-eval-v13 were used to evaluate the performance.

We experimented with various versions of the training data to discover the best performing TBL system setup. Between the boundary prediction models considered, the integrated HMM outperformed the joint tree model due to the HMM's tendency to make low-recall, high-precision decisions. Predicting different types of IP events improved filler detection accuracy, and it also led to smaller edit region prediction error rates on the integrated HMM. Regardless of the boundary event setup, TBL systems trained with predicted boundary events showed higher edit and filler detection accuracy on the test data.

Our post-processing strategy of ignoring edit region detection without corresponding IP predictions or IP predictions without detected edit regions did not cause performance degradation in edit region detection but instead resulted in slight improvements in error rates. TBL systems also generally improved detection of IPs before fillers. The improved filler IP predictions and the IP post-processing led to lower error rates in IP detection.

It is possible that the TBL systems might have fared better with the STT transcripts if their performance had been optimized on the STT transcripts of the development data. Modifying the objective function to evaluate scores for transformation rules on an STT transcript could be one way to tune the performance of the TBL systems for the STT condition, but it remains to be seen whether such approach would be effective.

The analysis of the TBL systems' theoretical performance on the test data with true boundary events suggests that significant gains in edit and filler detection performance can be achieved by improving the accuracy of boundary event predictions. Boundary events also provide an effective way to reflect prosodic information in edit and filler detection.

The TBL systems relied on POS tags, word identity information, and pattern match features to identify words in edit regions. Although those features are useful when the edit region is relatively short or when a pattern of repetition exist, they are ineffective for long edit regions and restart edit disfluencies. It would be necessary to use other lexical cues that can indicate deeper syntactic structure of utterances, such as features generated from a parser.

# Chapter 6

# CONCLUSION

Spontaneous speech typically contains disfluencies and conversational fillers that tend to make transcripts less readable than written text. Moreover, transcripts output by current STT systems are unsegmented, and this further limits the usability of automatic transcripts of spontaneous speech. In this thesis, we experimented with methods to detect sentence boundaries, disfluencies and conversational fillers in spontaneous speech transcripts without human involvement.

Detecting sentence boundaries can be thought as a slightly different problem than detecting edit regions and fillers. While it is natural to perform classification on each word boundary for sentence boundary or SU detection, it is necessary to classify the words for edit and filler detection. However, making classification decisions on word boundaries and words themselves are inherently related, since knowing the type of a word boundary would help identify edit regions and fillers. Therefore our overall system had a two-stage architecture. In the first stage, boundary events were predicted, and the boundary event predictions were then used in the second stage in which edit regions and fillers were identified.

Much research which has indicated that prosodic and lexical cues exist at locations of IPs and SU boundaries. We trained decision trees with a variety of prosodic and lexical features to predict boundary events. We also utilized the HE-LM to model word and boundary event sequences. Our experiments with decision trees showed that a combination of prosodic and lexical features yielded the best boundary detection performance. Although the decision tree trained with only the prosodic features achieved some success in SU detection, it failed to detect most IP events. Inclusion of lexical features improved detection performance substanstially, but detecting IPs, especially those IPs that occur after edit regions, remained difficult. The HE-LM demonstrated improvements in SU detection error rates over the decision tree model, but recall rates of IP events were lower with the HE-LM. Moreover,

the performance of the HE-LM degraded more severely than the decision tree model when the HE-LM was used to predict boundary events in the STT transcripts.

The HE-LM's higher performance on SU detection indicates that the word identity information is an important cue for SU detection. However, the HE-LM does not utilize pattern match information, which led to lower recall rates of IPs that occur after edit regions. Furthermore, the use of prosodic features in decision tree training lessened the effect of word recoginition errors in STT transcripts.

We explored three different model combination approaches to build classifiers that take advantage of the strengths of both the HE-LM and decision tree model. We observed improvements in error rates from all three model combination methods. Although overall prediction accuracies achieved with the combination models were similar, their characteristics were different. The predictions made by the joint tree model were relatively high-recall and low-precision. The recall rates of boundary events were the lowest with integrated HMM, but their precision was the highest. The independent combination model's prediction characteristic in terms of precision and recall fell mostly between the joint tree model and the integrated HMM. In addition, the detection performance of the indepedent combination model and the integrated HMM could be optimized for the STT trancripts by tuning component model weighting factor. Since the joint tree model could not be adjusted similarly for the STT transcipts, it suffered more performance loss with the STT transcripts.

We built classification models to predict different groups of boundary events. Experimental results showed that SU prediction accuracy is higher when the classifier is trained to predict both SU and IP events. However, due to co-occurrences of SU and IP events, the detection performance of IP events was lower under the joint SU and IP prediction paradigm. Using a new boundary event category that represents the co-occurence of SU and IP events resolved this issue. We also trained classifiers to predict different types of IPs. Although boundary event detection performance showed little change due to having multiple IP categories, we observed that the IP type information was useful for edit and filler detection.

We trained TBL systems to detect edit regions and conversational fillers. The TBL systems utilize lexical features such as words, part-of-speech (POS) tags, and pattern match

cues in addition to boundary events for edit and filler detection. Experimental results showed that identifying edit regions is a challenging task. Although filler detection performance was encouraging, the detection performance of edit regions left much to be desired. Improvements in both edit and filler detection performance were observed when the TBL systems were trained with predicted boundary events. Predicting the types of IP events led to lower edit and filler error rates in all system configurations except for edit detection with boundary events predicted by the joint tree model. Overall error rates were lower when the integrated HMM was used to predict boundary events.

An oracle experiment on performance achievable with the TBL systems indicated that increasing accuracy of boundary event predictions would result in substantial gain in edit and filler detection accuracy, particularly with multiple IPs. One area of improvement in boundary event detection is the prosody model. Even though research suggest that prosody plays major role in edit disfluency detection by humans (Lickley, Shillcock & Bard 1991), prosodic features used in our work was not very effective for IP detection. Modeling prosody with a parameterized statistical model such as Gaussian mixtures may lead to better performance.

Another possibility of improving IP detection could be found in the word fragments. Although word fragments almost always indicate occurrences of edit disfluencies, we did not use any features based on word fragments in our classifier models, since such features could not be generated automatically. If word fragments can be recognized automatically through advances in STT technologies or acousting processing, it could greatly improve IP detection performance.

Finally, we believe that utilizing information about syntactic structures of utterances has potential to improve both boundary event and edit region detection. Although such information is already reflected in our models through POS tags, the structural information remains highly localized and shallow. Incorporating syntactic information obtained from a parser into decision tree models or utilizing a parser to make metadata predictions (Hindle 1983) could lead to higher metadata detection performance.

# BIBLIOGRAPHY

BBN (2003*a*), *Rich Transcription Evaluation Framework using rteval v2.3*, BBN Technologies.
*http://www.speech.bbn.com/ears/rteval_v2.3.zip

BBN (2003*b*), *System description of BBN+ primary CTS STT system for RT-03 Spring evaluation - system B1*, BBN Technologies. Available from NIST.
*http://www.nist.gov/speech/tests/rt/rt2003/spring/

Bear, J., Dowding, J. & Shriberg, E. (1992), Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog, *in* 'Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics', Newark, DE, pp. 56–63.

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), *Classification and Regression Trees*, Wadsworth & Brooks Press, Belmont, CA.

Brill, E. (1995), 'Transformation-based error-driven learning and natural language: A case study in part of speech tagging', *Computational Linguistics* **21**(4), 543–565.

Buntine, W. & Caruana, R. (1991), *Introduction to IND and Recursive Partitioning*, NASA Ames Research Center. TR. FIA-91-28.

Charniak, E. & Johnson, M. (2001), Edit detection and parsing for transcribed speech, *in* 'Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics', Pittsburgh, PA, pp. 118–126.

Christensen, H., Gotoh, Y. & Renals, S. (2001), Punctuation annotation using statistical prosody models, *in* 'ISCA Prosody-2001', Red Bank, NJ. paper 6.

Godfrey, J. J., Holliman, E. C. & McDaniel, J. (1992), Switchboard: Telephone speech corpus for research and development, *in* 'Proceedings IEEE Conference on Acoustics, Speech and Signal Processing', Vol. 1, San Francisco, CA, pp. 517–520.

Heeman, P. A., Loken-Kim, K. & Allen, J. F. (1996), Combining the detection and correction of speech repairs, *in* 'Proceedings of the Fourth International Conference on Spoken Language Processing', Vol. 1, Philadelphia, PA, pp. 362–365.

Hindle, D. (1983), Deterministic parsing of syntactic nonfluencies, *in* 'Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics', Cambridge, MA, pp. 123–128.

Huang, J. & Zweig, G. (2002), Maximum entropy model for punctuation annotation from speech, *in* 'Proceedings of the Seventh International Conference on Spoken Language Processing', Denver, CO.

Jones, D., Wolf, F., Gibson, E., Williams, E., Fedorenko, E., Reynolds, D. & Zissman, M. (2003), Measuring the readability of automatic speech-to-text transcripts, *in* 'Proceedings of the Eighth European Conference on Speech Communication and Technology', Geneva, Switzerland, pp. 1585–1588.

Kim, J.-H. & Woodland, P. (2000), A rule-based named entity recognition system for speech input, *in* 'Proceedings of the International Conference on Spoken Language Processing', Beijing, China, pp. 512–524.

Kim, J.-H. & Woodland, P. (2001), The use of prosody in a combined system for punctuation generation and speech recognition, *in* 'Proceedings of the Seventh European Conference on Speech Communication and Technology', Geneva, Switzerland, pp. 2757–2760.

Kneser, R. & Ney, H. (1995), Improved backing-off for mgram language modeling, *in* 'Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing', Detroit, MI, pp. 181–184.

Levelt, W. & Cutler, A. (1983), 'Prosodic marking in speech repair', *Journal of Semantics* pp. 2:205–217.

Lickley, R., Shillcock, R. & Bard, E. (1991), Processing disfluent speech: How and when are disfluencies found?, *in* 'Proceedings of the Second European Conference on Speech Communication and Technology', Vol. 3, Genoa, pp. 1499–1502.

Liu, D. & Kubala, F. (2003), A cross-channel modeling approach for automatic segmentation of conversational telephone speech, *in* 'Proceedings IEEE Automatic Speech Recognition and Understanding Workshop', US Virgin Islands, pp. 333–338.

Liu, Y., Shriberg, E. & Stolcke, A. (2003), Automatic disfluency identification in conversational speech using multiple knowledge sources, *in* 'Proceedings of the Eighth European Conference on Speech Communication and Technology', Geneva, Switzerland, pp. 957–960.

Mangu, L. & Brill, E. (1997), Automatic rule acquisition for spelling correction, *in* 'Proceedings of the 14th International Conference on Machine Learning', Nashville, TN, pp. 187–194.

Mangu, L. & Padmanabhan, M. (2001), Error corrective mechanisms for speech recognition, *in* 'Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing', Vol. 1, Salt Lake City, UT, pp. 29–32.

Nakatani, C. & Hirschberg, J. (1994), 'A corpus-based study of repair cues in spontaneous speech', *Journal of the Acoustical Society of America* **95(3)**, 1603–1616.

NIST (2003*a*), *The Rich Transcription Fall 2003 (RT-03F) evaluation plan*, NIST. *http://www.nist.gov/speech/tests/rt/rt2003/fall/docs/rt03-fall-eval-plan-v9.pdf

NIST (2003*b*), *Rich Transcription Fall 2003 Workshop*, NIST. *http://www.nist.gov/speech/tests/rt/rt2003/fall/presentations/index.htm

Palmer, D. & Hearst, M. (1994), Adaptive sentence boundary disambiguation, *in* 'Proceedings of the 1994 Conference on Applied Natural Language Processing', Stutgart, Germany, pp. 78–83.

Ratnaparkhi, A. (1996), A maximum entropy part-of-speech tagger, *in* 'Proceedings of the Empirical Methods in Natural Language Processing Conference', University of Pennsylvania, pp. 133–141.

Reynar, J. & Ratnaparkhi, A. (1997), A maximum entropy approach to identifying sentence boundaries, *in* 'Proceedings of the Fifth Conference on Applied Natural Language Processing', Washington, D.C., pp. 16–19.

Shriberg, E. (1994), Preliminaries to a theory of speech disfluencies, PhD thesis, Department of Psychology, University of California, Berkeley, CA.

Shriberg, E. (1999), Phonetic consequences of speech disfluency, *in* 'Proceedings of the International Congress of Phonetic Sciences', Vol. I, San Francisco, CA, pp. 619–622.

Shriberg, E., Bates, R. & Stolcke, A. (1997), A prosody-only decision-tree model for disfluency detection, *in* 'Proceedings of the Fifth European Conference on Speech Communication and Technology', Rhodes, Greece, pp. 2383–2386.

Shriberg, E., Stolcke, A., Hakkani-Tür, D. & Tür, G. (2000), 'Prosody-based automatic segmentation of speech into sentences and topics', *Speech Communication* **32(1-2)**, 127–154.

Sönmez, K., Shriberg, E., Heck, L. & Weintraub, M. (1998), Modeling dynamic prosodic variation for speaker verification, *in* 'Proceedings of the Fifth International Conference on Spoken Language Processing', Vol. 7, Sydney, Australia, pp. 3189–3192.

Stolcke, A. (2002), SRILM - an extensible language modeling toolkit, *in* 'Proceedings of the Seventh International Conference on Spoken Language Processing', Denver, CO, pp. 901–904.

Stolcke, A. & Shriberg, E. (1996), Automatic linguistic segmentation of conversational speech, *in* 'Proceedings of the Fourth International Conference on Spoken Language Processing', Vol. 2, Philadelphia, PA, pp. 1005–1008.

Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Hakkani, D., Plauche, M., Tür, G. & Lu, Y. (1998), Automatic detection of sentence boundaries and disfluencies based on recognized words, *in* 'Proceedings of the Fifth International Conference on Spoken Language Processing', Vol. 5, Sydney, Australia, pp. 2247–2250.

Strassel, S. (2003), *Simple Metadata Annotation Specification Version 5.0*, LDC. *http://www.ldc.upenn.edu/Projects/MDE/Guidelines/SimpleMDE_V5.0.pdf