Error-Responsive Feedback Mechanisms for Speech Recognizers

Lin Lawrance Chase

April 11, 1997

Abstract

This thesis is about modeling, analyzing, and predicting errorful behavior in large vocabulary continuous speech recognition systems. Because today's state-of-the-art recognizers are not designed to be situated naturally in an error feedback loop, they are ill-positioned for inclusion in multi-modal interfaces, multi-media databases, and other interesting applications. I make improvements to the current approach to predicting and analyzing error behaviors, which is currently based only on the measurement of word error rate.

The speech recognizer's functionality is extended to include confidence annotations, which are "meta-level" markings that indicate how certain the recognizer is that it has decoded its input correctly. This is accomplished by feeding externally defined error conditions back to the recognizer. Error feedback enables the construction of statistical models that map measurements of the recognizer's internal states and behaviors to externally defined error conditions.

The measuring and modeling techniques used for confidence annotation are extended to create a blame assignment system for utterances whose actual transcripts are known. Errors are classified into a set of categories, some of which are directly useful in automatic adaptation schemes while others are more suited for human interpretation.

This classification approach is enhanced when used in conjunction with a visual error analysis tool that was developed during the thesis project.

Contents

List of Figures

List of Tables

Acknowledgements

The one person without whom this project never could have happened is Raj Reddy. He was a co-founder of the Robotics Institute, which was my first employer as a computer scientist (at the ripe age of nineteen), and has been my home for the past eight years. He was the thesis advisor of my first advisor, and my advisor for this project. He has always provided me with incredibly valuable support: sending me to conferences, giving me access to enormous computing capacity, introducing me to people, and always sending me the message that he knows I can do whatever needs to be done. From him I learned that having outrageously high goals can be a very good thing. I also learned important lessons by watching how he brings people together to build a loyal and hardworking group. His influence on my view of computer science and work in general will surely prove to be indelible.

One of the best things Raj ever did for me was ask me to work with Wayne Ward. Wayne has spent countless hours with me, teaching me the details and ins and outs of our end of the research business. In retrospect, his patience with me during my "wandering years" seems worthy of some kind of prize, if not canonization. I still aspire to be as good as Wayne at both deep thought and pure hacking power – maybe someday. But the most important thing I learned from Wayne is that it's possible to work hard and be really good, but at the same time maintain excellent good humor and an appreciation for the finer things in life.

While Raj made this project possible, it never would have been finished without the influence of Roni Rosenfeld. His attention to detail and willingness to give me the time and support I needed during one of the busiest years of his life combined to finally push me through to the finish. He thinks I did this project pretty much on my own, but in my heart I know he was absolutely essential. Both as a friend and lately as an advisor, Roni is one of the most real and caring people I have ever known. I feel really lucky to have worked with him. I'm also really glad he was there to answer the phone the night my sister had her baby – who else could call her advisor at 1 a.m. for help in deciding whether to go to the C-section? My time with Roni's family has been happy and fun – another invaluable gift to me.

Mei-Yuh Hwang taught me what it means to be truly serious and devoted to clear thinking. Her mathematical and programming abilities are astounding. I wish every girl who ever thought "math class is hard" could see her working for just ten minutes. In a committee member, friend, and work companion I couldn't have asked for more. (By the way, her kids are incredibly cute. Don't let that serious exterior fool you, she's the world's best mother.)

Jaime Carbonell has always been supportive and helpful. He may not remember this, but when I told him ten years ago that I wanted to become a technical writer, he told me in no uncertain terms that I should get a Ph.D. in "the real thing". And so I did – with him on my committee. Many thanks to Jaime for steering me right and giving me good feedback and direction.

In the final phases of the thesis work I finally had the chance to work with Alex Waibel and his group. I'm really glad I did – some of my most interesting collaborations came about in this context. In particular, Michael Finke and Torsten "Zeus" Zeppenfeld were excellent partners. Many thanks to Alex, Michael, and Torsten for all the time they spent with me. And many thanks to Alex for his final comments on the thesis, which definitely improved the work.

Eric Thayer and Ravi Mosur have both been pricelessly helpful during my years in the Sphinx group. They each must have answered at least a hundred phone calls and suffered scores of "just dropping in to ask a question" visits...never mind the mounds of email. I have learned everything from how to mount disks to how to train HMMs from these guys, and will never forget how cheerfully helpful they have been throughout.

Pierre Dupont helped with many aspects of the creation of this document. Stan Chen provided very useful and detailed feedback. Rich Stern, Pedro Moreno, Bhiksha Raj, Matt Siegler, Evandro Gouvea and all of the guys in "Rich's group" have always been friendly, helpful, and supportive. Paul Placeway always knows the answer to my most hackiest questions, and threw a great wedding to boot. Kristie Seymore has been an excellent influence on me – she's quick as a whip and really knows how to be happy. Alex Hauptmann taught me how to be a good experimentalist. Ken Goldberg taught me how to keep good logs and my head on straight at the same time.

Michael Witbrock provided a shoulder to lean on at the last minute, when it really counted. Sunil Issar has given me a quiet but constant reminder that good work simply is. Chengxiang Lu showed me the value of persistence and smiles. Alex Rudnicky reminded me of the value of hands-on teaching.

The list really goes on and on...CMU has been a great place to work and study. The people on both the facilities and administrative staffs have always been there for me when I really needed them. There are too many names to list here, but if we had less dedicated people CMU would not be the special place that it is.

The outside world has brought me much as well. Herb Gish, Marie Mateer, and Rich Schwartz at BBN have been great. Andreas Stolcke, Liz Shriberg, and Mitch Weintraub at SRI were all helpful and supportive. Bill Byrne, Fred Jelinek, Kimberly Shiring, and the whole staff at JHU have at times been saviors for me. Mari Ostendorf has often given me helpful feedback. The researchers at NSA and NIST have been great to work with, especially Barb Wheatley and Cal Alano, who were very supportive in the final phases. I feel very lucky to have worked in such a strong and friendly community.

In the end, though what I really want to say is this:

This work is dedicated to all the tigers in the world, especially the big one at home. Rar!

Chapter 1

Introduction

Imagine that you have recently purchased a large new luxury automobile, complete with all the usual controls: a steering wheel, gear shift, brake and accelerator pedals, etc. In the middle of the dashboard there is exactly one dial: a large and prominently displayed speedometer, marked in miles per hour. This is an excellent car that you've bought; it's very well engineered, finely detailed and fully equipped. At first it seems the perfect dream machine. But after some use you notice that the only thing the car can tell you about itself is how fast it's going. When the car is driven in new condition and good weather on well-paved roads, the speedometer is a useful feedback device. But get the car stuck in a ditch, or run out of fuel, or get low on transmission fluid, or cross the border to Canada, and suddenly that prominently displayed speedometer is no longer quite so useful.

Such is life in the speech recognition community today, where word error rate (WER) looms large and clear, front and center, as the only commonly used metric of system performance and environmental interaction. As a gold standard for measuring system performance in well understood and carefully controlled environments, WER is a useful tool. But move the microphone from close to the mouth of an office worker to the cockpit of a traveling vehicle, or imbed the recognition system in a multi-modal computing interface, and suddenly it becomes apparent that other (related!) tools and measures are needed.

Some drivers of automobiles literally only need a speedometer – "It won't go. Come fix it!" Others are comfortable with a bit more detail and can readily interpret fuel gauges and engine temperature sensors. More sophisticated owners are comfortable with oil dip sticks and timing lights. Some people, mostly trained mechanics, regularly use the extremely expensive and training-intensive diagnostic equipment typically found in dealers' garages. Automobiles are complex systems – to integrate them into society we've, over time, had to work out this progression of sophistication and training in car users. Both the complexity of vehicles and the changing nature of what exactly a car does in its environment have dictated exactly how feedback to the user has been handled.

This thesis is an attempt to advance the sophistication of the same sort of feedback process for complex speech recognition systems. Most state-of-the-art recognizers only give users the equivalent of that one little dial in the middle of an otherwise empty dashboard. I'd like to hand out the blueprints for a tachometer, fuel gauge, temperature gauge, and collision avoidance warning device.[1]

I will approach this goal through the development of a method of modeling, analyzing, and predicting error behaviors. The causes of error in a large vocabulary speech recognition system such as Sphinx-II or Janus fall roughly into one of a few categories: problems with dictionary pronunciations, inaccuracy in the acoustic models, inaccuracy in the language model, search errors, and interactions between the component acoustic and language scoring facilities. I build a set of computational units that perform as instrumentation of these aspects of speech recognizer behavior. These "instrument readings" are then used in two main models:

- a system that annotates recognizer output with markings that indicate how sure the recognizer is that its output is correct, and
- a system that, given a correct transcript of what was said, labels the possible origin of the errors made by the recognizer.

The initial chapters of this document are devoted to explaining the experimental environments in which this work was chiefly done.[2] One important goal in this project was to make the techniques developed and lessons learned appropriate for any system that shares basic design characteristics with the best contemporary systems. To insure that this goal was met, two of the best recognizers and two of the hardest data sets were chosen as testbeds. Chapter 2 describes the Sphinx-II and Janus speech recognizers. Chapter

[1]The air bags might have to wait a bit.

[2]Error analysis and early blame assignment work has been done on a total of four systems: Sphinx-II, Sphinx-III, Janus, and the commercial version of HTK.

3 describes the North American Business News (NAB) corpus, a collection of read news articles. This chapter also describes the Switchboard (SWB) corpus, a collection of recorded telephone conversations.

Chapter 4 contains a detailed motivational argument for using an error analysis approach that is more sophisticated than the "lone dial" word error rate (WER) model mentioned above. In this chapter examples are drawn using a software package[3] developed during the course of the thesis.

Chapter 5 describes the basics of building and evaluating confidence annotations. It also lays out baselines against which all confidence annotation results in the thesis are compared.

Chapter 6 describes a word-level confidence annotation experiment in which Sphinx-II is used to decode the NAB data. Chapter 8 describes the same type of experiment performed with Janus on the Switchboard data. In both of these experiments we will look in detail at the set of "instrument readings", or predictor variables, used to make confidence annotations. This is followed by a detailed analysis of combinations of these predictors, together with how they perform under a standard domain-independent metric. In each case we also discuss applications of the confidence annotations and how to evaluate their usefulness in particular application programs. A comparison of the two experiments is given in the final chapter.

The Chapter 6 and 8 experiments focus on predicting how well the entire recognizers is performing in making correct decodings. But the same sort of technique can be applied to looking at just the performance of the acoustic modeling that drives recognition. In Chapter 7 we look at a phone confidence annotator that indicates how correct the acoustic scoring mechanism is likely to be at any point in time. These experiments were performed using Sphinx-II on the NAB data, as an extension to the work in Chapter 6.

Chapter 9 describes a blame assignment approach that uses both the "instrument readings" from the confidence annotation work and correct transcripts of utterances to label errors as belonging to one of the categories:

- out-of-vocabulary (OOV) word spoken,
- search error,
- homophone substitution,

[3]The Visual Error Region Analysis (visERA) package is available through public ftp, and is also known as the CMU Error Analysis Toolkit.

- language model overwhelming correct acoustics,
- transcript/pronunciation problems in dictionary,
- confused acoustic models, or
- miscellaneous/not possible to categorize.

Some of these categorizations can be directly useful in automatic adaptation schemes. Others are more suited for analysis by the human system designers.

Chapter 10 summarizes the lessons learned in the course of this thesis, of which there are several types. The confidence annotation experiments across two systems and task domains are compared and contrasted. The usefulness of the blame assignment approach and its relation to the confidence annotation work is discussed. Observed behaviors of the speech recognizers that seem to lead to errors are summarized with an eye toward leveraging improvements in system design. And finally, a list of engineering problems encountered among a variety of speech recognizers are presented together with guidelines for how to avoid them in future systems.

Chapter 2

Speech Recognizers and Related Computations

This thesis describes techniques that are generally useful for error feedback on most of the state-of-the-art systems currently in use for large vocabulary speech recognition. Experimental work that demonstrates the usefulness of the approach developed here has been carried out on two such systems. The first, Sphinx-II, is a very large vocabulary continuous speech recognition system that uses semi-continuous Hidden Markov Model (SCHMM) acoustic models for decoding. The second, Janus, is similar in design but relies on continuous (CHMM) models for its acoustic scoring. In this chapter we describe the design and functionality of the major components of these systems, breaking out into parallel discussions where major differences occur.

Together these two systems cover most of the major design features of the best large vocabulary speech recognizers in use today [18] [50] [19] [36] [39] [40] [42] [43] [44] [48] [64] [63] [62] [16]. All of the experimental techniques described in this thesis have been designed to be portable to all such systems. During the development of these techniques tests and experiments were performed on a total of four systems: Sphinx-II, Sphinx-III [50], Janus, and the commercial version of HTK[1]. The two systems finally reported here were the most straightforward to use for engineering and other logistic reasons. The availability and stability of source code for the recognizers is currently of critical importance for this type of work. This is not an ideal state of affairs, however. For this reason a portion of Chapter 10 is devoted to a discussion

[1]See http://www.entropics.com/.

of how to improve the system engineering of recognizers to support the development of error feedback mechanisms such as confidence annotators and error blame assigners.

In this chapter I will give details on both the forced alignment and the decoding strategies employed by Sphinx-II [50] [27] and Janus [66] [14], as well as some additional side computations that are useful in building error feedback mechanisms. The descriptions of Sphinx-II and Janus focus mainly on those aspects of the systems which are measured and used as inputs to confidence annotation and blame assignment experiments described in later chapters.

2.1 Front End Signal Processing

The first step in both the Sphinx-II and Janus systems involve "front end" signal processing steps, as shown in Figures 2.1 and 2.2. This section describes how an uttered sentence moves from vibrating air through a microphone to a digitized form ready for recognition.

2.1.1 Sphinx-II Front End

The NAB data (see Chapter 3), used exclusively in this thesis with the Sphinx-II recognizer, was collected using Sennheiser close-talking microphones, which are mounted in front of talkers' mouths with headsets. The signals collected from the microphone are sampled at 16kHz. This bandwidth is often referred to as "full bandwidth" sampling for human speech. Each sample is in linear format with 16 bits per sample. The sampled waveform is then partitioned into 25.6 msec frames, which are spaced at 10 msec intervals. A pre-emphasis filter is applied, and 12 mel-scale frequency coefficients are computed for each frame [11] [56]. Four parallel sets of features are then computed from the MFC values.

1. Twelve cepstral values normalized by subtraction of the utterance mean,

2. 40 msec and 80 msec difference (Δ) MFCs, for an additional twelve features,

3. second order difference (Δ^2) MFCs, adding twelve more features,

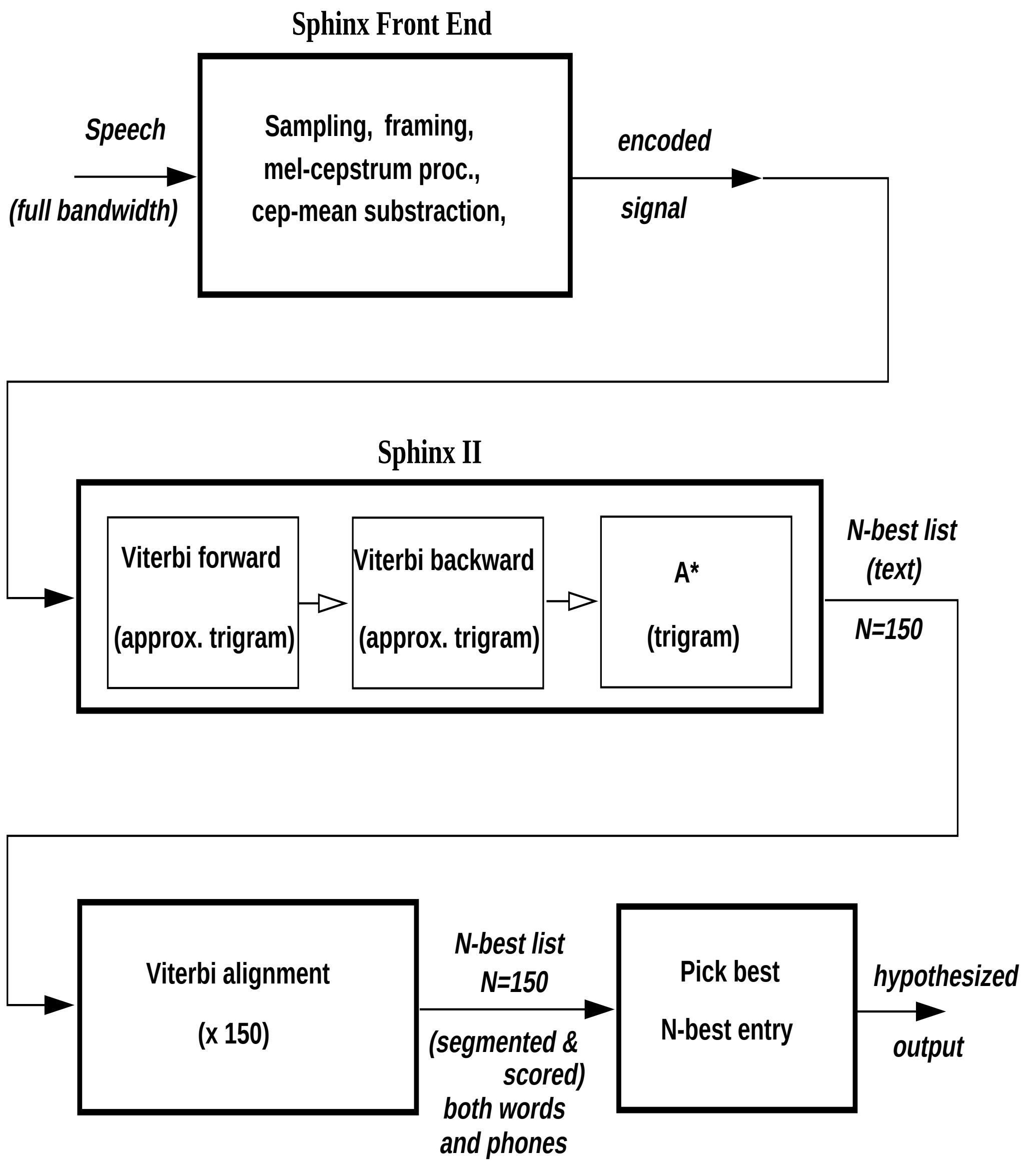


Figure 2.1: The complete speech recognition process based on Sphinx-II produces N-best lists of length 150 with complete phone segmentations and scores within each word. All NAB experiments are performed with this system.

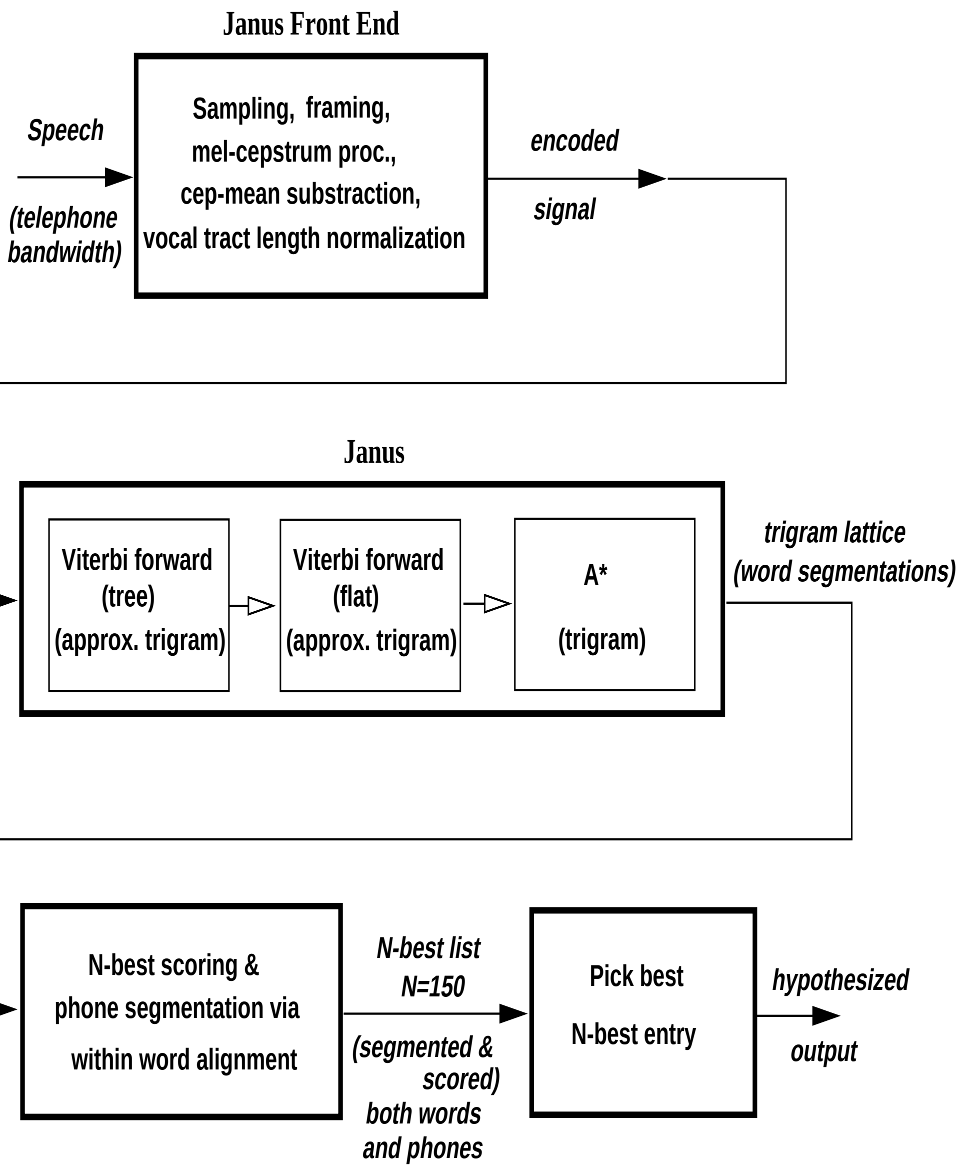


Figure 2.2: The complete speech recognition process based on Janus produces N-best lists of length 150 with complete phone segmentations and scores within each word. All Switchboard experiments are performed with this system.

4. log energy, difference (Δ) power, and second order (Δ^2) power, adding the final three features.

Thus in total thirty-nine features are computed for each frame. These features are assumed to be independent of each other, and are treated as four parallel, equally-weighted "streams" in the acoustic scoring module of Sphinx-II.

2.1.2 Janus Front End

The Switchboard data used with the Janus system is collected using standard telephone handsets with carbon-button microphones, yielding data that resides in the standard telephone frequency bandwidth. The sampling rate for Switchboard is 8000 samples per second. The data is sampled in μ-law format with 8 bits per sample. Frames at 10 msec intervals are produced, as in the Sphinx-II system. Twelve mel-scale frequency coefficients are computed at each frame. A total of thirteen cepstral coefficients (the original twelve plus one power) are produced, whose averages and corresponding power features yield 39 total features as above. In Janus all of these features are treated in a single stream. Cepstral mean subtraction is used, as is the vocal tract length normalization scheme described in [13]. Finally, linear discriminant analysis (LDA) is used to reduce the 39 features to a set of 24 features.

2.2 Acoustic Modeling

In this section we summarize the techniques used to transform the systems' acoustic representations into the search space characterizations that support the main recognition process.

2.2.1 The Phone Sets

Both Sphinx-II and Janus rely on a fundamental set of *basephones*, or context-independent phones, which are used both as the basis of dictionary word pronunciations and as the conceptual building blocks of the more sophisticated context-dependent models.

In Figures 2.3 and 2.5 we see a list of the basephones used in each system. (These figures also include phonological descriptions of each basephone which are described in Section 2.8.3.)

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
+INHALE+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+SMACK+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AA	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AE	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AH	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
AO	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AW	1	0	1	0	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0
AX	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AXR	1	0	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AY	1	0	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
B	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
BD	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
CH	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0
D	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
DD	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0
DH	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0
DX	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1
EH	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
ER	1	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
EY	1	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0
F	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0
G	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
GD	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
HH	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
IH	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
IX	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
IY	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
JH	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0
K	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
KD	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
L	1	1	1	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
M	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
N	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0
NG	0	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
OW	1	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0
OY	1	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0
P	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
PD	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
R	1	1	1	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
S	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0
SH	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	0
T	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
TD	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0
TH	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0
TS	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0
UH	1	0	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
UW	1	0	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
V	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0
W	0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0
Y	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
Z	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0
ZH	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	0	0
SIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SILb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
SILe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 2.3: Withgott and Chen's phonological descriptors for the Sphinx-II basephone set.

a	vocalic
b	consonantal
c	rhotic
d	advanced
e	high
f	low
g	back
h	rounded
i	tense
j	voiced
k	w-offglide
l	coronal
m	anterior
n	nasal
o	lateral
p	continuant
q	syllabic
r	flap
s	sonorant
t	front
u	y-offglide
v	distributed
w	strident
x	silence

Figure 2.4: Vector elements of the Withgott and Chen phonological descriptor.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
+UH+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+BR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+HU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+MB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+NH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+SM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+TH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
+LA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
AA	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AE	1	0	1	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AH	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
&AH	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
AO	1	0	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AW	1	0	1	0	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0
AX	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AXR	1	0	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
AY	1	0	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
B	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
BD	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0
CH	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0	0
D	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
DD	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0
DH	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0
DX	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1
EH	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
EN	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1	0	0
ER	1	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
EY	1	0	1	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0
F	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0
G	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
GD	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
HH	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
&HH	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
IH	1	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
IX	1	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
IY	1	0	1	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0
JH	0	1	0	0	0	0	1	0	0	0	1	1	0	0	1	0	1	0	0	0	1	0	0	0
K	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
KD	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
L	1	1	1	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0
M	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
&M	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0
N	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	0	0	0
NG	0	1	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0
OW	1	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0
&OW	1	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0
OY	1	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0
P	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
PD	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
R	1	1	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0
S	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0
SH	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1	1	0	0	0
T	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
TD	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0
TH	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0
TS	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0
UH	1	0	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
UW	1	0	1	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0
V	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0
W	0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0
Y	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0
Z	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0
ZH	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	1	0	0	1	1	0	0	0
SIL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 2.5: Withgott and Chen's phonological descriptors of the basephones used in the SWB experiments with Janus.

Both systems model deleted stops (consonants found at the end of words which are pronounced differently than their beginning- and mid-word counterparts) such as "TD" and "DD". Both systems also model the schwa "AX" and its companion "AXR", which is the schwa followed by a clear R sound.

The phone sets differ minorly in how they model silence: Sphinx-II uses the "SIL" model for all between-word silences, "SILb" for beginning-of-utterance silences, and "SILe" for end-of-utterance silences. Janus, on the other hand uses only "SIL" for all of these purposes.

Noise models for discrete sounds such as lip smacks, coughs, and door slams are used in both systems. They are labeled with tokens that begin and end with a "+". There are more of these models in Janus, primarily because conversational speech contains a wider variety and many more training examples per utterance of this class of sounds.

The Janus system models the "syllabic R", represented by "ER". Sphinx-II does not use an explicit model for syllabic R's. Neither system models the syllabic L or M, but Janus does model the syllabic N.

The Janus system includes a few additional phones whose symbols begin with the symbol "&". These phones, such as "&AH" and "&M", are used only in the dictionary pronunciations of conversational interjections such as "HMM" and "UH-HUH". They are intended to capture the differences between "normal" pronunciation of these phones and the longer, prosodically different versions of the phones found in the "filler words".

2.2.2 Sphinx-II Semi-continuous Modeling

The Sphinx-II system relies on semi-continuous HMMs which are built with a fixed five-state Bakis topology. The phone models are context-dependent triphones which are further distinguished by their position within words as either single-phone, begin-word phone, mid-word phone, or end-word phone. The set of triphones modeled is made up of all of the within-word and cross-word triphones possible given the dictionary in question. For the NAB experiments reported here the total number of triphones modeled is 84,580. Each triphone is modeled with a set of five senones[28]. Triphones are clustered to enable sharing of distributions (and thus training data) at the senone level. The clustering is done in two phases. The first is a top-down splitting algorithm that creates a decision tree using linguistically motivated questions to create splits in conjunction with an entropy measure. After the tree is fully grown it is pruned from the leaves toward the root to a fixed number of leaf nodes (and thus senones). During pruning only nodes with the highest value of maximum change in entropy are retained. Clustering is permitted only within the same state location of the same base phone. (Thus clustering is done within phone and within state position within phone.) For these experiments 10,000 senones were used to model the 84,580 triphones. After clustering this yielded a set of 21,716 unique senone sequences (of length five), which yields a factor of four reduction in the number of free parameters which need to be estimated.

The semi-continuous modeling relies on 256 codebook entries (or mixture components). (For the NAB experiments reported here no phone-class dependent codebooks were used, although this functionality is supported in Sphinx-II.) Training of the mixture components (or codebooks) is initialized using k-means clustering of the cepstral vectors. The Baum-Welch algorithm [4] is used to estimate both output and transition probabilities from this point, which is initially specified with uniform output distributions. For the experiment reported here the acoustic training data set was divided into two parts – one for male speakers and the other for female. One set of acoustic models was trained for each partition. Thus the triphone clustering is the same for all experiments, but the output and transition probabilities differ depending on gender of the test speaker.

2.2.3 Janus's Polyphonic Modeling

Acoustic modeling in Janus is quite distinct from that of the Sphinx-II system. The fundamental modeling unit is not the triphone but the 5-phone. Janus's method of clustering codebooks and distributions yields a system that in the end is somewhere between semi-continuous and continuous on the modeling spectrum.

Three-state model topologies are used for all 5-phones. Training of the 5-phone models is initialized with a set of context-independent codebooks. This means that, for the training of the Switchboard models used here, initially there are approximately 500,000 distributions (one for each unique 5-phone model) which are modeled with $3 \times N$ codebooks, where N is the number of basephones. A set of speaker-dependent training labels is produced using the MLLR algorithm. An iteration of training then proceed, which includes a round of LDA, k-means estimation (under about 4000 samples) of the initial codebook means, and then 3 or 4 iterations of the EM algorithm for estimation of the distribution covariances (which are diagonalized), output probabilities, and model transition probabilities. After this initial round of training the first top-down clustering of codebooks (which are each of length 16) is performed. Minimum counts of 1000 samples are enforced in this codebook clustering. For the models used here a total of 6000 codebooks were produced. After codebook clustering a round of distribution clustering is performed in a greedy bottom-up fashion based on an entropy measure. Minimum counts of 200 samples are enforced here, and a total of 24,000 distribution clusters are created. After this process has been completed, we return to the speaker-adapted labeling step and repeat the training process, initializing from our new 24k/6k position.

This parameter tying scheme is unique to Janus, and appears to be one of the system's strongest design characteristics. The system computes up to 24,000 distributions at each frame during search. (The actual number computed is typically smaller due to pruning.) This makes it very difficult to keep performance statistics at the most detailed phone model level. Thus, as described later, we typically will collect statistics on phone behaviors at the basephone level of abstraction, including the word-position information of single phone, begin phone, middle phone, or end phone.

2.3 The Dictionaries

A dictionary is a list of known vocabulary words together with one or more pronunciations for each word. The pronunciations are described using the system's set of basephone models.

Two dictionaries are used in this work. The first, used with Sphinx-II for a set of experiments on read North American Business (NAB) news data, contains 19,980 unique words and a total of 29,066 pronunciations. On average each word in the dictionary has 1.4 pronunciations. Most words (86%) actually only have one pronunciation, while 2,868 (14%) of the words have two or more pronunciations. The pronunciations for the dictionary used in the NAB experiments are derived from a wide variety of sources. Many pronunciations are the results of hand-editing performed during the years of development of the Sphinx-II system on the WSJ and NAB data sets.

The second dictionary, used with Janus for a set of experiments on the Switchboard data set, contains 14,162 word entries, for which there are a total of 15,601 pronunciations. On average each word has 1.1 pronunciations. Most words (91%) have only one pronunciation. 1,359 words (9%) have two or more pronunciations. The pronunciations used in the Switchboard dictionary are derived mainly from the PRONLEX dictionary[2].

2.4 The Language Models

In both sets of experiments reported here, trigram language models are used. Different smoothing and pruning approaches, are used, however. Statistics on both are presented next.

2.4.1 The NAB Language Model

The smoothing algorithm for the construction of the NAB trigram language model is widely referred to as the "Katz" approach, and is described in [31]. The NAB language model was constructed using the toolkit described in [54], which is a widely adopted standard package for building Katz N-gram language models.

The run-time calculation of language model probabilities from a Katz trigram language model can be expressed as one of five cases. In all of the NAB experiments reported here a tag that denotes the origin of the language model probability within these cases is produced and carried along with the actual probability. Throughout this document the following symbols are used to denote the five cases:

1. *T:* The trigram (w_1, w_2, w_3) is present in the language model database.
2. *BB-B:* The bigram (w_1, w_2) is present and the bigram (w_2, w_3) is present.
3. *B:* Only the bigram (w_2, w_3) is present.
4. *BB-BU:* Only the bigram (w_1, w_2) is present.
5. *BU:* Neither (w_1, w_2) nor (w_2, w_3) is present.

The language model used in the NAB experiments was trained on 227 million words of NAB-style text data. Bigram and trigram cutoffs of 1 and 3 respectively were used. This produced a language model that contains 19,980 unigrams, 5,027,012 bigrams, and 6,703,115 trigrams.

[2]See http://www.ldc.upenn.edu/.

2.4.2 The Switchboard Language Model

The Switchboard language model was trained on an order of magnitude less data, only 3,048,775 words. The smoothing algorithm used in the Janus system for language models is the Kneser-Ney backoff scheme [32]. The language model used contained a total of 14,150 unigrams, 401,402 bigrams, and 1,313,567 trigrams. No cutoffs were used in its construction.

2.5 Search

Both Sphinx-II and Janus rely fundamentally on left-to-right time-synchronized Viterbi decoding embedded in a beam search as their main method of search control. Both systems are also three-pass systems which rely heavily on detailed acoustic processing in the first two passes and on an A* algorithm in the third. This, however, is where the similarities end. Each system is described separately below.

In both cases individual word scores are created by combining:

1. a word acoustic score, $P_{AC}(x|w)$, in which x is the acoustic segmentation in question, with

2. a word language model probability, $P_{LM}(w|h)$, in which h is a history of context words.[3]

The combination is calculated at the word level in the following manner:

$$P(w,x) = P_{AC}(x|w) \times P_{LM}(w|h)^{LW}$$

where LW is a global constant called a "language weight" that is set empirically. The use of LW is the method by which the score contributions from the language model are balanced with those from the acoustic models.

The path score of a sequence of words is actually calculated in *log* probability space. A path score in *logProb* space is calculated as follows:

$$logP(w_1..w_n, x) = \sum_{i=1}^{n} logP_{AC}(x_i|w_i) + (LW \times logP_{LM}(w_i|h)) + IP$$

[3]In all of the experiments reported here, we model the history h by collapsing all histories that end with the same two words into one class. Thus every prediction is based on a two-word history. This type of language model is called a "trigram" language model, as discussed above.

where IP is a constant "insertion penalty" which is applied to each word in the final path sequence of length n. [4] The log base used varies according to the recognizer's implementation.

2.5.1 Sphinx-II's Three Pass Search

The Sphinx-II system used in this thesis work, described thoroughly in [23], [24], and [26], is a three-pass system. The first two passes use Semi-Continuous Hidden Markov Models (SCHMMs) based on 10,000 senones and a trigram language model to produce a very large set of possible word segmentations. The third pass, which supports a wide variety of language models and is based on an A* search formalism, generates ordered lists of possible decodings based on the acoustic segmentations created by the first two passes.

The First Two Passes The first pass of the system is a time-synchronous Viterbi search that produces a tree of word possibilities rooted at the start frame of an utterance. Each word segmentation (start frame, end frame, wordID triple) present in the output of the first pass of the system has a start frame that is temporally contiguous with the end frame of its predecessor word. Thus each word segmentation present in the output of the first pass is linked via a plausible path to the start node of the utterance. Not all word segmentations present in the output of the first pass have a follower word, however, due to pruning. This means that many end nodes of the data structure created by the first pass do not correspond to the end of the utterance.

Each of the word segmentations present in the acoustic output of the first pass have an associated acoustic score which is calculated directly from the SCHMMs. A trigram language model is used to guide the search, but the scores are only stored locally, as they are used for pruning but not final scoring, which is handled in the third pass.

The second pass of the system is a backwards time-synchronous Viterbi search. This pass is more constrained than the first, however, as it is guided by the word segmentations present in the first pass output tree. For each

[4]In general we decode paths which start with a minimum-length silence word $w_0 =< s >$, and which end with an analagous $w_n =< /s >$. Typically we assume that $P(< s >) = 1$ when computing the total path score.

word segmentation present in the first pass output, the second pass, while moving backward, will generate additional possible start times for words and will also possibly find additional possible predecessor words, drawing from among those words present in the first pass tree. As in the first pass, the acoustic scores produced during this pass are calculated directly from the SCHMMs.

The first two passes produce a large set of possible word segmentations whose start and end times were generated directly by either the first or second pass of the system. In addition, we have in the abstract the cross product, for each word instance, of all its start and end times, which creates a much larger set of word segmentation possibilities. Acoustic scores for most of these cross possibilities are not directly available and so must be estimated from pre-calculated scores of temporally similar segmentations actually produced by one or both of the first two passes. A detailed description of the averaging mechanism used for this purpose can be found in [1]. In general the system performs better in terms of WER when it is permitted to choose from among this greatly expanded set of word segmentations, even though the acoustic scores used were not computed directly from the SCHMMs.

When a particular word is spoken it will typically give rise to scores or even hundreds of distinct segmentations. The search algorithm that produces these segmentations is very generous in allowing words to begin well before their optimal entry point and to linger well beyond their optimal exit. Since the recognizer is designed to account for every frame of data in the input stream using some word from its vocabulary, this flexibility is a necessary feature that in the end makes the system relatively robust to noise, verbal stumbles, and other generally hard-to-match sounds.

The Third Pass After the first two passes have completed, word segmentations present in the output of the two passes are linked backward and forward to their previous/next best-scoring neighbor(s). [5] Thus the topology of the graph created for word sequences can be considered to be a "bigram topology". Each word segmentation that does have a connection to a previous word also has an associated language model score[6] that corresponds to

[5]Due to pruning during search a given segmentation is not guaranteed to have both forward and backward links.

[6]Throughout this document the term "language model score" should be taken to mean the language model probability together with the language weight and word insertion penalty. In any place that describes the actual language model probability before combi-

the path its left-hand neighbor belongs to. Typically this score is a trigram score that looks back two words in the left-hand path for context. Because storing all of the possible trigram scores associated with each word segmentation is very costly in space, only the best trigram score for any given word segmentation is stored. Thus the language model scores used to generate the results of the first two passes can be described as "trigram scores computed on a bigram topology" of word links.

The third pass of the Sphinx-II system (also described in [1]), is an A* search that starts at the beginning of the utterance and generates an N-best list of paths, which are reported as word sequences ordered by likelihood under the acoustic and language models in question. It uses the output of the first two passes of the system, regenerating language model scores, but relying on exact and estimated acoustic scores from the first two paths, as described above. This A* pass optimizes the selection of start and end frames for a given word segmentation carefully, and is built to support more sophisticated language models than the trigram. The trigram, however, is the language model used exclusively in this work.

The third pass produces an ordered (by total path score) list of decodings of the utterance in question. The decodings consist of sequences of word tokens, including "hidden" silences and noise words. For the experiments reported here typically 150 ranked decodings were reported by the system.

2.5.2 Search in Janus

Janus's first pass of search is a left-to-right Viterbi that is similar to Sphinx-II's first pass. The same "trigram scoring on a bigram topology" is employed. What results from the first pass is a large graph that contains many possible start frames for each word that survived the beam pruning. This pass is called the "tree" pass because its search is controlled with the use of a lexical tree, in which words that share the same initial phones are searched at the same time. In the second pass, which in Janus is also left-to-right, as opposed to right-to-left as in Sphinx, is called a "flat" pass. In this pass all words sequences that were not pruned after the first pass are explored, but no lexical tree is used. This pass also relies on the suboptimal "trigram scoring on a bigram topology". The third pass is an A* search as in Sphinx-II. It is

nation with these other weighting factors, the term "language model probability" will be used.

optimal with respect to the trigram language model. One major difference between the systems is that no acoustic estimation is used in Janus's third pass – only actual acoustic scores from the flat (second) pass are used. The Janus system can dump the lattice traversed in the third pass for future use. (Sphinx-II is capable of this, too, but the Janus lattices are one to two orders of magnitude smaller, making this a much less daunting proposition when running large experiments.) As described in Figure 2.2, in the Switchboard experiments these lattices were used in a later phase to create N-best lists that contained word and phone segmentations and scores.

2.6 Acoustic Alignment

The fundamental idea of this thesis is that showing the recognizer examples of the mistakes it makes can allow it to create and use a model of its own error behaviors. In order to define what errors are we need some appropriate definition of "ground truth". This ground truth is the basis for deciding what is and is not correct output for a recognizer under some set of circumstances, and is therefore a key underpinning idea.

2.6.1 What is The Right Answer?

What is the "right answer" for the output of a recognizer on a given utterance? This is a question that begs many others. For one thing, the best answer can be very application specific. A system that "listens" to telephone conversations and tries to get the gist of what's being said for automatic reporting purposes will have very different "right answers" than a system that is intended to support detailed transcription of the same telephone conversations.

On another note, the best answer requires taking a careful look at what the component knowledge sources are in the recognizer, together with a list of exactly which levels of control and abstraction are going to be accessible and useful. A gisting system might only care about the detection of certain important content words with a healthy ratio of detections to false alarms, while a dictation system will rely heavily on accurate and precise frame-level segmentation decisions. Thus we see that ground truth can be specified at very different levels depending on the goal of the system.

Finding the best answer to this question requires consideration of the total amount of human time available for the definition of truth. In general, labeling of spoken language data is quite expensive. The total cost depends in part on on the type and quality of labels required. A gisting system may require only a few labels for each conversation of several minutes' length. Reference transcripts for dictated materials can be arbitrarily detailed – the word sequence may or may not be exact, phonological and prosodic labels may or may not be included, and associations of words or phones with time stamps may or may not be included.

2.6.2 Types of Data and the Ground Truth

One main source of data used in this thesis draws on read speech from adult speakers, most of whom are native speakers of North American English. The database contains dictation on an utterance-by-utterance basis of texts drawn over a period of years from the Wall Street Journal and similar newspapers, all of which report business activities related to North American concerns. The database is called the "North American Business News" (NAB) corpus.

Compared to many speech recognition problems, this sort of data is "well-behaved". For instance, speakers in the datasets generally pronounce words according to expectations that are derived from standard dictionaries. Only one person is speaking at any given time. The acoustic data is sampled so that it is not limited with respect to the normal human auditory bandwidth. Also, the environment in which the data was collected, a quiet office situation, is relatively benign. Most of the non-speech noise in the data is generated by the speakers themselves by breathing, smacking of lips, or accidental brushes against the microphone, which is mounted via a close-talk headset near the mouth. In sum, these data characteristics work together to create utterances which are generally continuous sequences of relatively clearly pronounced words, generally uninterrupted by "unreasonable" sounds.

While in general the data is "well-behaved", it must be noted that there are very challenging aspects to this task. The difficulties become apparent when considering the nature of the speech and the (related) goal of the task. Compared with other domains the vocabulary, the syntactic and semantic coverage of the material, and the phonological space covered are all enormous. Most importantly, the goal of the task is to have an automatic system produce a *perfect* word-level transcript of each utterance.

Considering the characteristics of the speech data and the goals of the task, it is apparent that one good "right answer" for this body of data is a very accurate word-level transcription of each utterance, made and checked by a human listener. Indeed the dataset as distributed includes such transcripts. Alignment of recognizer output (at the word level) against these transcripts forms the basis of the word-level scores that are the standard measure of performance in this domain, word error rate (WER).

Unfortunately it is not clear that the word-level transcriptions used together with the forced alignment approach provide a good basis of "ground truth" for the conversational speech data in the Switchboard database. However, this approach is the best method currently available, as very little phonetically transcribed data has been produced to date. Thus the same approach is used on both data sets in this thesis.

2.6.3 Producing Alignments

The ground truth alignment of words and their component phones to the frames of an acoustic utterance is accomplished via a constrained Viterbi search. The required inputs to the process are:

- A *dictionary* that maps word strings to sequences of context-independent phone pronunciations. Multiple pronunciations, entered as separate lines in the dictionary, are supported. (As described below, the aligner will pick the best-scoring pronunciation from among the multiple entries.)

- A set of *acoustic models* that are identical to those used in the recognition tests which will be compared against the "ground truth" alignments.

- A complete set of word-level *reference transcripts*. Each word that appears in the reference transcripts must also appear in the dictionary. Words in the transcripts that have multiple pronunciations in the dictionary will be aligned against all possible pronunciations unless a specific pronunciation is noted in the transcript. In general this is not the practice – the aligner is typically asked to find the best pronunciation for a word.

- A *control file* that indicates the exact set of acoustic utterances and their reference transcripts for a given set.

Given this information, the aligner steps through each entry in the control file. For each entry the reference transcript is accessed. The words in each transcript are "exploded" into their component context-independent phones via table lookup in the dictionary. A graph of context-dependent triphones is then created that represents the pronunciation of the reference transcript. (This graph will be flat everywhere except within and adjacent to words with multiple pronunciations, where branching will occur in proportion to the number of alternative pronunciations.) Between words in the graph optional silence and noise models are typically allowed. [7] This graph then becomes the constraining search space for a left-to-right Viterbi alignment of the acoustic frames of the utterance against the acoustic models. At the completion of the Viterbi computation for the last frame of the utterance the aligner dumps the optimal state, phone, and word assignments to frames, along with the actual scores associated with each segmentation.

An example output of such an alignment can be seen in Figure 2.6. For each word aligned, the optimal start and end frames under the acoustic models used are reported, along with the total acoustic score associated with the segmentation. The component phones of the words are reported in exactly the same fashion. (The start frame of the first phone of a word corresponds exactly with the start frame of the word. The final phone and word end frames correspond exactly as well.) For the remainder of this document, this description of alignment between words, phones, and acoustic data will serve as our definition of ground truth for utterances.

2.7 Creating Phone-detailed N-best Lists

Both Sphinx-II and Janus produce results by calculating and saving word-level acoustic scores, which are combined with language model scores to produce path scores. Because the recognizers are implemented with extensive space and time optimizations that support real time computation, their internal data structures are not set up to report the necessary phone-level segmentation and score information. In particular, it is not possible to get the recognizers to produce detailed phone segmentation and score information for the words they output in the course of the acoustic search phases of their recognition runs.

[7] In some systems, such as Sphinx-II, noise words (such as those that model lip smacks) must appear in the reference transcript.

To compensate for this in Sphinx-II, we add an additional step of processing to the recognizer, in which each element of the (textual) N-best list output for an utterance is aligned against the acoustic models to produce phone segmentations and scores within words. In the Sphinx-II experiments (on the NAB data) this alignment is constrained only at the start and end frame of the utterance. In the Janus experiments (on Switchboard) we do the alignment within the lattices, constraining the start and end frame of each word in turn as we produce N-best lists. (The N-best lists in both the Switchboard and NAB experiments were of length 150).

Figures 2.1 and 2.2 depict the complete overall design of each of the two decoding systems used in this thesis.

2.8 Useful Side Computations (ν)

Error analysis based only on the recognizer-internal state information available from the recognition systems described above is inherently limited, as recognizers are built to make optimal use of the their knowledge sources. Chapter 5 includes empirical results that show exactly how much error-predictive "squeezing" we can do from this starting point.

To support better analysis and error prediction work we can turn to a variety of supplemental computations. Some of these are truly independent of the knowledge of ground truth. The remainder can be trained successfully on a modest amount of development data.

2.8.1 Phone-only Decoding

It is often useful to compare the typical word-level decodings with alternative phone-level decodings. A "phone-only" decoding is not constrained by the dictionary or language model – any basephone (including silence and noise models) can follow any other without regard to dictionary pronunciations or word sequence preferences from the language model. Within phones, however, the state sequences imposed by the acoustic model topologies are respected, as are any minimum phone duration constraints. Phone decodings are generally most accurate when some sort of language-specific phone sequence model is used to constrain the search. (This phone sequence model is analagous to a word-level language model but is trained at the phone level of abstraction.)

For the NAB experiments reported in this thesis a bigram phone model trained on a dictionary with more than 100,000 entries was used as the phone sequence model. The phone decodings were performed using context-dependent (triphone) acoustic models.[8] The resulting phone accuracy on the NAB data sets varied at 72-73%.

No phone-only recognizer was available for the Switchboard/Janus experiments.

2.8.2 Best Acoustic Scores by Frame

Another useful side computation is made by looking at the best possible completely unconstrained acoustic score at each frame, together with the basephone identity of the model that produces it. In the Sphinx-II system used for the NAB experiments this involved looking at the best-scoring senone at each frame. (Each frame is treated independently.) In the Janus system this amounted to identifying the best-scoring model at each frame. In both cases there is a unique mapping from the best-scoring acoustic model to a single basephone. The information saved for use in the experiments reported here is a report of the best basephone and best score at each frame.

Because Sphinx-II uses semi-continuous models, every senone's value is computed at every frame during the first pass of the Viterbi search.[9] Thus it is possible to get a best score for each basephone, not just the top-scoring basephone, at each frame, at no additional cost. This means that for the NAB experiments we have at our disposal an additional side source of information: a matrix of best scores for each basephone at each utterance frame.

Getting an analagous complete matrix out of the Janus system would be extremely costly due to the very large number of acoustic models in the system.[10] However, the top-"few"-scoring states and their corresponding basephones are available at minimal cost. In the Switchboard experiments I elected to used only the best-scoring basephone and its score at each frame.

[8]The acoustic models used in the NAB experiments for phone decoding were not the same as in the word decodings. Thus the actual scores computed during phone decoding were not directly comparable with the acoustic scores of the word decodings. This problem did not arise in the Switchboard experiments, as no phone-only decoder was available for Janus.

[9]There are 256 senones in the system used for these experiments.

[10]Compare the upper bound of 24,000 models computed at each frame in Janus to the 256 senones in Sphinx-II.

2.8.3 Comparing Basephones (ν)

In this work it is often useful to be able to compare one basephone with another, asking questions such as, "How similar are these two phones?", and "How confusable are these two phones?" The experiments that follow refer frequently to the two comparison measures described next. The first is based solely on a phonological model of the phonemes represented by the basephone set. The other is an empirical model of basephone confusability that was developed from direct observation of the decoding systems used.

Distance in W/C Phone Space In [61], Withgott and Chen describe a 24-element vector of phoneme attributes that create a space in which our basephones can be distinguished from each other[17]. The attributes in this space include features that help separate vowels from consonants, front vowels from back vowels, and liquids and glides from fricatives, etc. The complete set of features and their (binary) values for each basephone are shown in Figures 2.3 and 2.5, with a key to the vector element identities given in Figure 2.4.

We use a simple Hamming distance to compare any two entries in one of these tables. This means that we compare each feature of the two phones separately. If they have the same (binary) value then the Hamming distance is not incremented. If they do not have the same value then the Hamming distance is incremented by one. Thus the minimum distance between two phones is zero, and the maximum is 24. Note that all silence and noise models have the same descriptor vector, thus $H_{WC}(SIL, +SMACK+) = 0$. Two similar vowels such as UW and IY produce a small value, $H_{WC}(UW, IY) = 3$. Two dissimilar basephones such as UW and F produce larger values, as in $H_{WC}(UW, F) = 11$.

In the work reported here, this distance metric is typically computed on a frame-by-frame basis when comparing the acoustic performance of a word-level decoding against some other standard.

Frame-Level Confusability A more empirical measure based on experienced phone confusions can also be used to measure the "distance" between phones.[11] To build this metric we compare the phone segmentations that come from word decodings with the phone segmentations that come from

[11]This confusion-based metric was not used in the Switchboard experiments.

force-aligning utterances against their reference transcripts. In both cases we use the segmentation information to create a frame-to-basephone mapping. The two resulting mappings (one for the REFerence and one for the recognizer HYPothesis) are then compared on a frame-by-frame basis. The statistics collected in this fashion are stored in a confusion table. (The confusion tables for the male and female NAB models are presented in Appendix A. This confusion metric was not used in the Switchboard experiments.)

Each confusion matrix is square and contains one row and one column for each basephone. (The entries in the matrix are asymmetric, as described below.) The column headers refer to the phone found in the HYPothesized words. The row labels refer to the basephone labels found in the REFerence forced alignments. The integer found at the entry $< Phone_{HYP}, Phone_{REF} >$ ($< col, row >$) represents the number of frames guessed as $Phone_{HYP}$ that were "really" $Phone_{REF}$, as defined by forced alignment. To use this table as a distance metric between two phones we must identify one of the phones as our HYPothesized phone and the other as the REFerence.[12] The confusion-based distance between two phones, $CD(Phone_{HYP}, Phone_{REF})$, is taken to be zero if $Phone_{HYP} = Phone_{REF}$. If $Phone_{HYP} \neq Phone_{REF}$ then we:

1. calculate the total number of frames in column $Phone_{HYP}$ not guessed correctly. This amounts to the sum of the column entries excluding $< Phone_{REF}, Phone_{REF} >$. Call this value $Wrong(Phone_{HYP})$, and

2. take the ratio $< Phone_{HYP}, Phone_{REF} > / Wrong(Phone_{HYP})$, which represents the proportion of wrong guesses of $Phone_{HYP}$ due to $Phone_{REF}$.

The confusion-based distance metric is thus zero between any phone and itself, and a fraction varying between zero and one for any other pair of phones.

2.9 Summary

This chapter discussed the designs of the two state-of-the-art large vocabulary speech recognizers which drive the experiments reported in later chapters. Both forced alignment of word-level transcripts and full-scale recognition

[12]For confidence annotation work, in which the true REFerence phone is not known, the REFerence phone labels must come from some sort of "best guess", such as a phone-only decoding.

were covered. Additional side computations, useful in building error feedback mechanisms, were also discussed.

```
Words
<s> 0 39 -5928957
SIL 40 56 -2917317
DREXEL'S 57 103 -6146159
MR. 104 136 -4379214
JOSEPH 137 181 -7224360
SIL 182 187 -1260122
CALLS 188 215 -3633419
STALEY'S 216 270 -7896969
SIL 271 295 -4496927
ALLEGATIONS 296 366 -10158174
SIL 367 373 -1453233
OUTRAGEOUS 374 462 -14586843
</s> 463 478 -2313649
Phones
SILb 0 39 -5928957
SIL 40 56 -2917317
D(SIL,R)b 57 62 -1005091
R(D,EH) 63 65 -480115
EH(R,K) 66 72 -896515
K(EH,S) 73 79 -910896
S(K,AX) 80 83 -555837
AX(S,L) 84 88 -584376
L(AX,Z) 89 94 -662125
Z(L,M)e 95 103 -1051204
M(Z,IH)b 104 110 -984861
IH(M,S) 111 114 -460485
S(IH,T) 115 122 -1189176
.
.
.
S(AX,SIL)e 436 462 -3926051
SILe 463 478 -2313649
END
```

Figure 2.6: Aligner output for the utterance "Drexel's Mr. Joseph calls Staley's allegations outrageous." Entries are: word, start frame, end frame, acoustic score.

Chapter 3

Experimental Data

The goal of this chapter is to present and describe the training data and test sets used in the experiments reported in later chapters. Specific attention is paid to issues that might relate to extending these techniques to other tasks and types of data and recognizers.

The first section describes the portions of the widely used North American Business News corpus [49] [46] [45] [47] [33] that were used for acoustic training, error feedback training, and independent test materials. This section describes the subset of the data used, and includes baseline statistics for the performance of the tools described in the previous chapter.

The second section gives a parallel description for the portion of the Switchboard data set used.

3.1 North American Business News Data

The North American Business News corpus (NAB), commonly referred to as the "Wall Street Journal" corpus, includes acoustic and language data collected between 1987 and 1995. It is described fully in corpus reports generated during those years [49] [33].

As described in Section 2.6.2, the corpus consists mainly of native speakers of North American English reading stories from the Wall Street Journal one sentence at a time. The database was collected in support of a wide variety of experiments done at many sites. The selection of the data used here was influenced by the following attributes of the database.

- *Microphone:* Almost all of the data in the corpus was collected using close-talking Sennheiser noise-cancelling microphones, mounted near the speakers' mouths via headsets. Much of the data was also collected in parallel under various alternative sampling systems, including a variety of microphones and telephone-bandwidth devices. Some corpus data included additive noise from environments such as car cockpits. For this work I have focused exclusively on the Sennheiser microphones, whose data are known colloquially as "channel one data".

- *Acoustic sampling:* For this work I have focused exclusively on the "clean" data, always working with "full bandwidth" speech. That is, the speech is sampled at 16 kHz in order to assure non-aliased representation of an auditory bandwidth that starts at 0 kHz and goes to 8 kHz. No articulatory phenomena that aid in the interpretation of speech are known to occur outside this 8 kHz bandwidth.

- *Acoustic environment:* Most of the data in the corpus was collected in relatively benign acoustic environments, such as quiet offices. No special selection from among the corpus components was necessary to maintain this characteristic. Most of the non-speech noises in the data come from the user in the form of breathing, lip smacks, and accidental motions against the microphone.

- *Number of speakers:* Portions of the corpus were designed to support work in speaker-independent recognition systems, while others were designed with adaptation to a specific speaker in mind. Both "many speaker" and "few speaker" sets of data are available in the corpus at large. I have focused on the "many speaker" subsets of the corpus, as the baseline system is a speaker-independent recognition system.

- *Amount of data from each speaker:* Commensurate with the "many speaker" and "few speaker" notions in the previous point, some portions of the database have a relatively small amount (order of 35 utterances) of speech per speaker, while others may contain up to an hour of speech for each of a small number of speakers. I have focused on the data subsets that contain a small amount of speech for each of many speakers.

- *Known vocabulary:* Some portions of the corpus were constrained to cover only words from a known vocabulary list. I did not use this data, as I wanted to approach and deal with "the unknown word problem", and thus did not want to have any assumptions about the vocabulary content of the data I used. In fact, however, the data in the corpus that is not restricted by vocabulary is still relatively well-behaved with respect to out-of-vocabulary (OOV) words when compared with a typical recognizer dictionary and language model. As reported in Table 3.1, words that are not in the system's vocabulary typically appear at or below a 3% rate.

- *Vocabulary size:* Through the years that the NAB corpus was collected the generally accepted standard of what was considered to be a "state-of-the-art" vocabulary size in this task domain grew dramatically from 5000 words to approximately 65,000 words.[1] The growth in vocabulary size occurred as an attempt to reduce overall error rates by reducing the OOV rate. This approach does have a direct impact on the overall error rate, but has a great cost in terms of both space and time of computation.[2] This reduction in OOV rate, and thus overall error rate, was accomplished by judicious and copius selection of vocabulary elements from language model training data. This thesis in part explores an alternative method of dealing with the OOV problem. Instead of making the problem go away by building a bigger language model, I am attempting to instrument the system so that it can guess where it might be faced with an error-making situation, including the OOV situation. For this reason a baseline vocabulary size of 20,000 words was chosen.

- *Standard use of the data:* Community standards have evolved around the use of the NAB corpus. Some of these standards were designed into the data, while others have arisen over time as generally accepted good practices. In all cases I have adhered to these standards, taking the strictest possible interpretations of them where there is any ambiguity. This list summarizes the main issues that arose in the design and execution of the experiments reported in this document.

[1]At the time that the core of the thesis work began (1994) the standard was 20,000 words.

[2]As a result, management of the considerable memory and computation time costs associated with using vocabularies and language models for 65,000 words have thus become the central system design issues in recent years.

  - *Acoustic training:* Exact lists of utterances are provided with the corpus for each type of acoustic training data supplied. I work exclusively with the data set called "si_284", which is a label that stands for "speaker-independent, 284 speakers". This data is read speech sampled from newspaper articles during the period 1987-1989, and for several years (until 1994) was the standard set of acoustic training utterances against which all competing systems were trained in controlled tests. There is no overlap between the content of these acoustic training utterances and the development data (see below) used for other training purposes in the system. See [49] for complete details. Section 3.1.1 below covers the exact use of this data and further characteristics of this data as they pertain to this work.
  - *Lexical pronunciations:* In general the modeling of word pronunciations has been left as a non-controlled parameter in the development of systems used on the NAB corpus. However, a few popular approaches have dominated activity in this area, so it is worth stating that a majority of the word pronunciations used in these experiments were drawn from Dragon System's large vocabulary dictionary [3]. Exact details are given in Section 2.3 below.
  - *Language model training:* As with acoustic training, a standard approach to language modeling has been developed for the purpose of controlled comparisons of system performance. A complete description of this process can be found in [54]. In 1994 [54] a standard trigram language model was constructed for shared use. It was derived from a standard body of language model training data and was based on a fixed 20,000 word vocabulary that was also derived from the setaside language model training data. Details of the language model and its performance on each of the data sets used in this work can be found in Section 2.4 below.

- *Types and Purposes of Data Sets:* Good research practice requires the complete separation of data used for this type of work into four distinct sets. See [30] for a good example of the correct use of the distinct types of data. This paper originates from the IBM speech research group, which is generally credited with convincing the rest of the community to adopt these standards in the use of data.

1. *Acoustic training data:* Recorded utterances which should share the acoustic, phonetic, and language characteristics of the target task domain but should be completely distinct in all ways (recordings, prompting text, and speaker identity) from all other data.

2. *Language training data:* Word-level transcripts that should share the language characteristics of the target task domain but should be distinct in content from all other data. With time-related materials such as news reports all language model training data should pre-date the following two subsets of data (development and evaluation test).

3. *Development test data:* Recorded utterances drawn from the task domain directly. Should be completely distinct in all ways (recordings, prompting text, and speaker identity) from acoustic training data and language training data. With time-related materials such as news reports, development test data should be preceded by all language training data in the calendar, and in turn should precede the evaluation test data. Used to develop and tune the recognition system, this data often becomes extremely familiar to system developers, and should therefore eventually become suspect as a source of true understanding of the nature of the problem.

   For this thesis work it is development test data that is used to identify and train useful error feedback mechanisms. For this reason the development test sets used in this work will sometimes be referred to as *error training data* sets.

4. *Evaluation test data:* Recorded utterances drawn from the task domain directly. Should be completely distinct in all ways (recordings, prompting text, and speaker identity) from acoustic training data, development data and language training data. With time-related materials this data is usually the most recent in time. It is generally considered bad form to use this set of data for more than a very occasional check on system performance.

   For this thesis work it is the evaluation test data that is used to measure the usefulness of the confidence annotation and blame assignment mechanisms developed.

3.1.1 The Acoustic Training Data

The acoustic training data used for the NAB experiments reported here was the standard si_284 set from the original Wall Street Journal corpus. The data was treated in two partitions, one for female speakers (of which there were 141) and another for male (143). This data set is sometimes called the "short term speaker" training set, as each speaker recorded relatively few utterances compared to an alternative "long term speaker" set. The short term set was used in this work in order to match the training condition as closely as possible with the testing conditions, in which many speakers contribute a few utterances to each set. In total 17,864 female utterances and 17,927 male utterances were used in training.

3.1.2 The Development Test/Error Training Data (ν)

The development test set used in the NAB experiments contains all of the full bandwidth, non-noise speech included in the official ARPA test sets between 1992 and 1995 for short term speakers. In total there are 1,523 utterances in this set. Table 3.1 summarizes some baseline statistics. With the approximately 20,000 word vocabulary used in the NAB experiments (see Section 2.3) the out-of-vocabulary word rate on these tests varies between 2.3% and 3.0%. Two baseline word error rate (WER) figures are quoted as well. The first, marked "original", refers to the initial error rate of the hypotheses output by the third (A*) pass of Sphinx-II. (See Figure 2.1.) The second, marked "N-best", refers to the error rate achieved by the entire system described in the same figure. Overall an the WER of the original system on the development data is 14.8%. After the realignment of the N-best lists the WER drops to 14.7%.

3.1.3 The Evaluation Test/Independent Test Data (ν)

The evaluation test set used in the NAB experiments contains all of the full bandwidth, non-noise speech included in the official ARPA test sets between 1992 and 1995 for short term speakers. In total there are 1,162 utterances in this set. Table 3.2 summarizes the corresponding baseline statistics. With the approximately 20,000 word vocabulary used in the NAB experiments (see Section 2.3) the out-of-vocabulary (OOV) word rate on these tests varies between 2.4% and 5.8%. Two baseline word error rate (WER) figures are

Test Set	Sex	#utts	%OOV	WER original	WER N-best
si_dev20	m	195	3.0%	11.8%	11.9%
si_dev20	f	205	2.9%	11.7%	11.6%
si_dt_20	m	246	2.5%	18.1%	17.5%
si_dt_20	f	257	2.3%	13.7%	13.9%
h1_dt_94	m	155	2.7%	18.4%	18.6%
h1_dt_94	f	155	2.7%	13.1%	13.2%
h3_dt_95	m	157	2.5%	15.8%	15.2%
h3_dt_95	f	153	2.9%	15.0%	14.7%

Table 3.1: Baseline statistics on the NAB development test sets.

Test Set	Sex	#utts	%OOV	WER original	WER N-best
si_evl20.nvp	m	169	3.5%	10.4%	10.4%
si_evl20.nvp	f	164	3.5%	11.0%	10.4%
si_et_h1	m	107	5.7%	18.3%	18.4%
si_et_h1	f	106	5.8%	12.0%	11.5%
h1_et_94	m	156	2.4%	15.8%	15.5%
h1_et_94	f	160	2.5%	15.5%	15.9%
h3_et_95	m	150	3.4%	20.2%	20.0%
h3_et_95	f	150	3.3%	16.1%	16.0%

Table 3.2: Baseline statistics on the NAB evaluation test sets.

quoted as well. The first, marked "original", refers to the initial error rate of the hypotheses output by the third (A*) pass of Sphinx-II. (See Figure 2.1.) The second, marked "N-best", refers to the error rate achieved by the entire system described in the same figure. Overall an the WER of the original system on the evaluation data is 15.0%. After the realignment of the N-best lists the WER drops to 14.9%.

3.2 Switchboard Data

The Switchboard database is a collection of telephone conversations between pairs of people who did not know each other prior to the recording session. The participants were given the option to talk on one of a large variety of topics. Some example topics are public education, drug testing, the federal budget, gardening, baseball, affirmative action, auto repairs, AIDS, air pollution, the use of credit cards, care for the elderly, recipes, cooking, football, music, Puerto Rican statehood, the Soviet Union, and immigration.

What follows are some paraphrased points from the Switchboard database documentation that indicate the nature of the data:

- *Collection control:* The Switchboard data was collected without human intervention, under computer control. Interaction between the two speakers and the system was via touchtones and recorded instructions. The two talkers, once connected, could warm up before recording began. Automation guarded against the intrusion of experimenter bias, and guaranteed a degree of uniformity throughout the long period of data collection. The prompting protocols were intended to elicit natural and spontaneous speech by the participants. The transcribers' ratings indicate that they perceived the conversations as highly natural.

- *Collection acoustics:* The use of T1 lines and automatic switching software made it possible to collect the digital version of the speech signals directly from the telephone network, and also to isolate the two sides of the conversations. The goal was to have real telephone speech, routed through the public network, but with no degradation due to the collection system. Isolation of the callers, within the limits of network echo cancelling performance, permits researchers to train on each speaker's voice separately, and then test on either one or both speakers in any conversation.

- *Speaker frequency:* Forty-eight people participated 20 times or more; this adds up to about an hour of speech for each of these active participants. Hundreds of others participated ten times or less. 542 speakers participated overall.

- *Speaker demographics:* The participants' demographics, as well as the dates, times, and other pertinent information about each phone call, are recorded in relational database tables. Except for personal information about the callers, these tables are included with the corpus. The volunteers who participated provided information relevant to studies of voice, dialect, and other aspects of speech style, including age, sex, education, current residence, and places of residence during formative years. It was intended that the talkers be broadly representative of adult speakers of American English between 20 and 60 years of age. From the beginning, a bias toward higher socioeconomic and educational levels was considered inevitable, due to the requirements of the task. It was also recognized that a serious effort would be required to insure representation of all dialects. The consent form signed by participants asked where the applicant grew up during the first 10 years of life. This community was then located on a wall map of the United States with the boundaries between the major dialect areas drawn on it. The names for these seven areas, plus the term "MIXED," were then used to classify each person by dialect. Because a large number of Texas Instrument employees, their relatives and acquaintances responded to calls for participation, there is a far greater number of "SOUTH MIDLAND" callers than would be expected, for example, in a random nationwide sample.

- *Prompts:* The prompts were devised with several criteria in mind. These included: covering many different topics of conversation, choosing subjects that interest large numbers of people, choosing subjects that tend to generate friendly differences of opinion or viewpoint, or invite exchanging of stories or shared experiences, and avoiding overlapping or subordinated topics and sensitive or personal issues as much as possible. The prompts were recorded by an experienced female speaker at 20 kHz, then downsampled appropriately.
- *Microphone:* Speakers used a variety of telephone handsets.
- *Acoustic sampling:* Telephone bandwidth, 8000 Hz sampling, 8-bit per sample μ-law encoding were the acoustic standards employed.
- *Acoustic environment:* About half of the Switchboard utterances are backchannel responses to the activity on the opposite paired channel. Crosstalk is present in some cases.
- *Number of speakers:* One person spoke on each channel.

3.2.1 The Acoustic Training Data

The Switchboard acoustic training data used here consists of approximately 70 hours of speech from each gender, for a total of 140 hours. For the experiments used here, gender-independent models were trained. Thus only one gender-independent system was run on both the development and evaluation data.

3.2.2 The Development Test/Error Training Data (ν)

The Switchboard data is divided into basic units called "conversation sides", which are each five minutes in length. These conversations sides are segmented into utterances that vary in length from one word to 200. Each conversation side generally has a unique speaker, thus error rates on this data are typically quoted per speaker. Tables 3.3 and 3.4 contain summary information for the development test set used in the Switchboard experiments reported here. (The data was broken out into "channel A" and "channel B" conversation sides strictly for convenience of presentation. The recording conditions for each paired conversation side were identical.) Overall the Switchboard development test set used contains 2,112 utterances. In the Janus-based experiments reported here, gender-dependent acoustic models were not used. For this reason the gender of the speaker for each conversation side is not noted. Overall the WER on the development test set is 36.1%.[3]

3.2.3 The Evaluation Test/Independent Test Data (ν)

Tables 3.5 and 3.6 contain summary information for the evaluation test set used in the Switchboard experiments reported here. Overall the Switchboard evaluation test set contains 1,384 utterances. In the experiments reported here, gender-dependent acoustic models were not used. For this reason the gender of the speaker for each conversation side is not noted. Overall the WER on the evaluation test set is 36.7%.

3.3 The Gender of Speakers

Gender (or anything highly correlated with gender, such as average fundamental frequency (F0)) is an important factor in accurate acoustic modeling. Gender modeling is usually accomplished via data splitting – all training

[3]These performance figures do not include any acoustic adaptation beyond initial training.

data is segregated based on a transcript or other independent labeling that identifies the gender of the speaker. This has the unfortunate effect of reducing the amount of training data available for training, typically by a factor of two. While various efforts are under way to eliminate this splitting problem [9], this simple approach was still the best available at the time the NAB experiments were run. This is a small problem for the NAB set used in this work because the size of the training set is relatively large and because the state sharing accomplished via senone modeling and the triphone decision trees work well enough to create "reuse" of many phone segments in the acoustic training data. It is a much bigger problem in the training of confidence annotators, however. In this case (as described in Chapters 5, 6, 7, and 8) the "training data" is actually the development test set, which is much smaller than the acoustic training set. To minimize any losses due to data splitting while realizing maximum recognition performance, in the NAB experiments I have used gender-specific acoustic models for decoding but recombined genders for training of confidence annotators. See Section 6.9 for details on how this is accomplished.

In the Switchboard/Janus experiments, MLLR acoustic adaptation can be used [35]. This makes it possible to avoid the use of (and thus avoid the negative data-splitting effects of) gender-specific models. The Switchboard experiments reported here were conducted using gender-independent acoustic models.

3.4 Summary

This chapter described the training data and test sets used in the experiments reported in later chapters. The rationale for selection of specific subsets of the two was covered. Baseline performance statistics were presented, and the data use guidelines respected throughout the experimental portion of this work were described.

Conversation Side	WER
sw2121-A	43.0%
sw2131-A	42.4%
sw2151-A	31.9%
sw2229-A	34.7%
sw2335-A	43.2%
sw2434-A	31.6%
sw2441-A	25.1%
sw2461-A	46.9%
sw2503-A	34.0%
sw2505-A	34.1%
sw2632-A	25.4%
sw2702-A	42.0%
sw2724-A	29.3%
sw2733-A	31.4%
sw2752-A	40.1%
sw2753-A	43.9%
sw2836-A	27.6%
sw2838-A	35.4%
sw3035-A	24.3%
sw3528-A	47.5%
sw3756-A	31.1%
sw3942-A	45.2%
sw3994-A	38.7%
sw4179-A	34.6%
sw4643-A	39.3%

Table 3.3: Baseline statistics on the Switchboard 1996 development test set, channel A.

Conversation Side	WER
sw2121-B	32.3%
sw2131-B	34.4%
sw2151-B	22.6%
sw2229-B	60.9%
sw2335-B	38.5%
sw2434-B	29.5%
sw2441-B	47.4%
sw2461-B	25.5%
sw2503-B	40.7%
sw2505-B	32.5%
sw2632-B	39.0%
sw2702-B	24.7%
sw2724-B	35.2%
sw2733-B	34.5%
sw2752-B	42.5%
sw2753-B	36.7%
sw2836-B	32.6%
sw2838-B	27.0%
sw3035-B	25.8%
sw3528-B	34.9%
sw3756-B	43.4%
sw3942-B	36.1%
sw3994-B	50.5%
sw4179-B	25.0%
sw4643-B	40.0%

Table 3.4: Baseline statistics on the Switchboard 1996 development test set, channel B.

Conversation Side	WER
sw3157-A	39.4%
sw3264-A	32.4%
sw3380-A	22.4%
sw3494-A	56.3%
sw3505-A	61.1%
sw3538-A	30.9%
sw3689-A	15.4%
sw3822-A	50.6%
sw3824-A	39.4%
sw3835-A	36.1%
sw3927-A	25.4%
sw3940-A	36.6%
sw4073-A	48.0%
sw4093-A	30.4%
sw4141-A	27.9%
sw4178-A	42.2%
sw4322-A	28.1%
sw4338-A	51.0%
sw4373-A	40.0%
sw4835-A	26.2%

Table 3.5: Baseline statistics on the Switchboard 1996 evaluation test set, channel A.

Conversation Side	WER
sw3157-B	26.6%
sw3264-B	46.4%
sw3380-B	25.3%
sw3494-B	49.0%
sw3505-B	26.2%
sw3538-B	37.2%
sw3689-B	49.3%
sw3822-B	59.3%
sw3824-B	34.3%
sw3835-B	32.0%
sw3927-B	57.0%
sw3940-B	51.9%
sw4073-B	38.2%
sw4093-B	32.2%
sw4141-B	35.4%
sw4178-B	28.2%
sw4322-B	26.4%
sw4338-B	30.7%
sw4373-B	26.4%
sw4835-B	49.1%

Table 3.6: Baseline statistics on the Switchboard 1996 evaluation test set, channel B.

Chapter 4

Motivation: Why WER Isn't Enough

The current practice of using word error rate (WER) as the standard metric of performance for speech recognition systems, while useful as a common currency and as a hard bottom line for performance measurement, is limited in its ability to provide insight into the behavior of complex recognition systems. The causes of error in a large vocabulary speech recognition system such as Sphinx-II or Janus fall roughly into one of five categories: problems with dictionary pronunciations, inaccuracy in the acoustic models, inaccuracy in the language model, search errors, and interactions between the component acoustic and language scoring facilities. If we look only at WER performance in evaluating our systems we will be unable to understand how each of the component models and their interactions contribute to the bottom line error performance. Also, WER captures only limited information about where on the time line errors occur and how they relate temporally to the correct decoding of the reference transcript.

In the example in Figure 4.1 the recognizer has output two words to accommodate one word in the transcript. The begin and end times of the error

```
REF: richard SARAZEN *** chief  ...
HYP: richard SIZE    AND chief  ...
```

Figure 4.1: Recognizer behavior for the OOV word "Sarazen".

Simple alignment produces:

```
REF: second month DR. SHENAUGH LEFT  FOR ARGENTINA ***********
HYP: second month DOG AND      SHEEN ALL EFFORT    ARGENTINA'S
```

Phonological alignment produces:

```
REF: second month DR. *** SHENAUGH LEFT  FOR    ARGENTINA
HYP: second month DOG AND SHEEN    ALL   EFFORT ARGENTINA'S
```

Figure 4.2: A problem with simple alignment that is corrected with phonological alignment.

actually correspond to the begin and end boundaries of the missed word in the input stream. While it may be "fair" to count two errors in this situation, it is not particularly helpful in analyzing the recognizer's behavior; nor is it likely that a high level summary of insertions, deletions, and substitutions that contain similar patterns will help us to understand how to modify the recognizer.

A simple step toward adjusting the WER metric for this time-related problem is to use phonological alignment (see Figure 4.2), in which the dictionary pronunciations of words are aligned at the basephone level, instead of aligning word tokens. Although in our example this adjustment does straighten out the incorrect labeling of an insertion and a substitution, the larger problem of understanding exactly what caused the error sequence in the first place remains. Also, it is very difficult to ascertain exactly how acoustic models fail (and how acoustic modeling failures interact with the language model's behavior) if we are constrained to analyze behavior at the word level.

With these issues in mind, and in keeping with the main theme of this thesis, a new error analysis technique (called "Error Region Analysis", or "ERA") has been developed. In this chapter we will examine how this new technique works, as well as the sorts of recognizer errors it can detect. Along the way we will also introduce some of the central ideas that form the basis of the error prediction work presented in later chapters will be introduced.

The error analysis approach discussed here has a few key characteristics:

1. The analysis is not done at the word level of abstraction, as is the case in other approaches, but rather at the frame level in the coded speech signal.

2. Since errors are not described at the word level, but instead at the frame level, the basic unit of analysis is no longer a word error (SUBstitution, INSertion, or DELetion), but an *error region*. An error region is a contiguous set of frames beginning and ending at word boundaries. It captures the location of recognizer errors in a fashion that allows us to look carefully at the interactions between language, lexical, and acoustic models.

Next we will look at exactly how these error regions are defined, how they are used to identify and analyze errors, and what the results of this analysis are on the data sets under investigation.

4.1 What is an error region? (ν)

An error region is a contiguous set of frames of acoustic data. It starts at the beginning of a word and ends at the end of a (possibly distinct) word. An error region is identified by comparing some ground truth definition (REF) of an utterance against some hypothesized (HYP) segmentation of the utterance. The ground truth is usually taken to be the forced alignment of the reference transcript against the acoustic data using the available acoustic models. [1] We normally analyze the recognizer's best guess, but the method described here can be used for any hypothesis.

In general, for each word segment in the reference the hypothesis must contain a segment with the identical word (including pronunciation variant), start frame, end frame, and thus acoustic score. If this matching criterion can be met within the hypothesis for all segments in the reference, then the hypothesis contains no error regions. If, however, when moving from left to right (beginning to end) in a reference definition, a non-match segment is discovered, then the begin point of an error region is identified.[2]

[1]This is intended as a cost-effective approximation to hand labeling, which is laborious and time-consuming.

[2]This is not the strictest possible definition of an error region start location. If the acoustic models used in the system are cross-word triphone models, we would in fact have to start the error region at the beginning of the final phone of the word preceding the start

After identifying a start point, the error region is extended word by word as the reference is traversed until a reference segment is found which again matches a segment in the hypothesis exactly. The location of the end of the error region is the end of the word that precedes this first rejoined match. It is possible that an error region may contain only one word. In the most extreme case, an error region will contain all the words in the utterance.

We can generalize this definition of *error region* in two ways:

1. A *frame tolerance* can be specified that allows some slack in the comparisons between boundary locations in the reference and hypothesis sequences. When comparing recognizer output to forced alignments of read speech, as in the NAB experiments reported later, this frame tolerance is usually two frames. This value is set as a function of the quality of the acoustic models applied to particular tasks. In the Switchboard experiments reported later in the thesis a frame tolerance of three is typically used, as the acoustic models for telephone conversational speech are not as good as those for full-bandwidth read speech.

2. The criterion for locating the end of an error region can be changed to reflect the effects of the language model being used by the recognizer. Consider the case of a word (left-to-right) bigram language model being used in conjunction with the above error region definition. The final word in the hypothesized error region may be different from its reference counterpart. This means that the language model scores of the word immediately following each may be impacted, as the errorful word in the hypothesis region will be a different context word for the following (correct) word than it is in the reference. (The only case in which the two distinct final words would play the same role in the language model would be if the hypothesized word were a pronunciation variant of the reference word.) Thus the language model score for the following

location described here. This would capture any possible differences in acoustic score that would result from guessing an incorrect basephone in the first position of the first word in the error region. For cross-word 5-phone acoustic modeling we would have to define the error region as starting at the beginning of the next-to-last phone of the preceding (correct) word. If the preceding (correct) word were a single phone word, this backward cascading might continue even farther back into the acoustic history. In practice these effects are relatively small. For simplicity these effects have been ignored in this thesis, and wherever error regions are used they are defined as described above.

(correct) word will be wrong, even though the recognizer's choice of the follower word was correct. To capture this effect for a trigram language model we must extend two words ahead instead of one.

For this reason a *window size* parameter is available that can be set to a value of 0, 1, or 2. These values correspond respectively to ignoring language model effects, assuming a bigram language model, and assuming a trigram language model. The parameter is interpreted as the number of successor words that should be included in the error region after the hypothesis returns to matching the reference.[3]

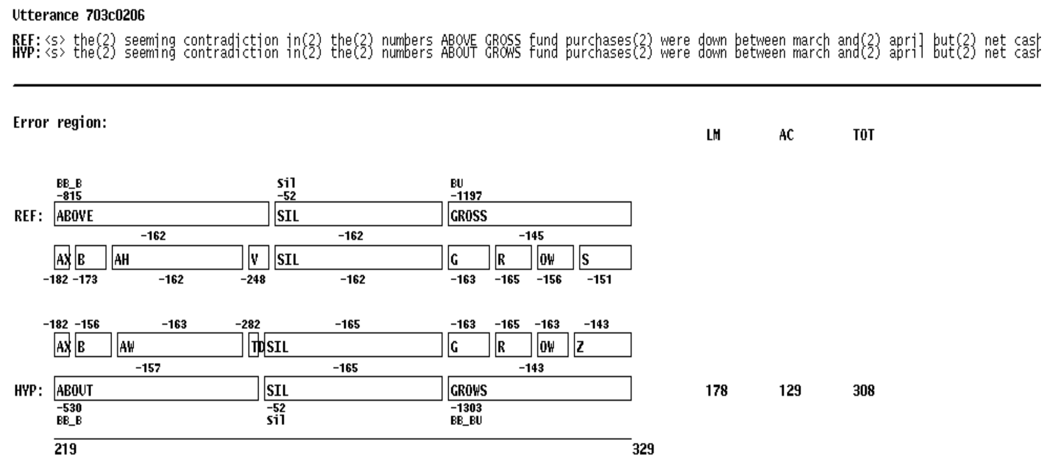
Naturally, more sophisticated language models may also affect error region definition. In contrast to short-term models such as the bigram and trigram, however, they can cause effects that are much harder to model. A trigger or cache language model[53], for instance, might by extension cause entire sequences of utterances to be considered as an error region, even though very few segments would have been considered as errorful under the word error rate (WER) metric. For this reason, the window size option, which allows us to ignore long term language model effects, is provided.

In Figures 4.3 through 4.7 we see examples of the same utterance analyzed using different values for the frame tolerance and window size. In the first example (Figure 4.3), the default values of frame tolerance (0) and window size (0) are used to analyze the utterance. At the top of the figure is the standard word alignment used to calculate the word error rate (WER) of the hypothesis. Two substitutions ("ABOUT" for "ABOVE" and "GROWS" for "GROSS") are found under the standard analysis, yielding a total of two word errors.

The error region analysis at the bottom of the figure reveals that the substitutions are not exact at the frame level, and that the problem is a combination of contributions from both the language and acoustic models.

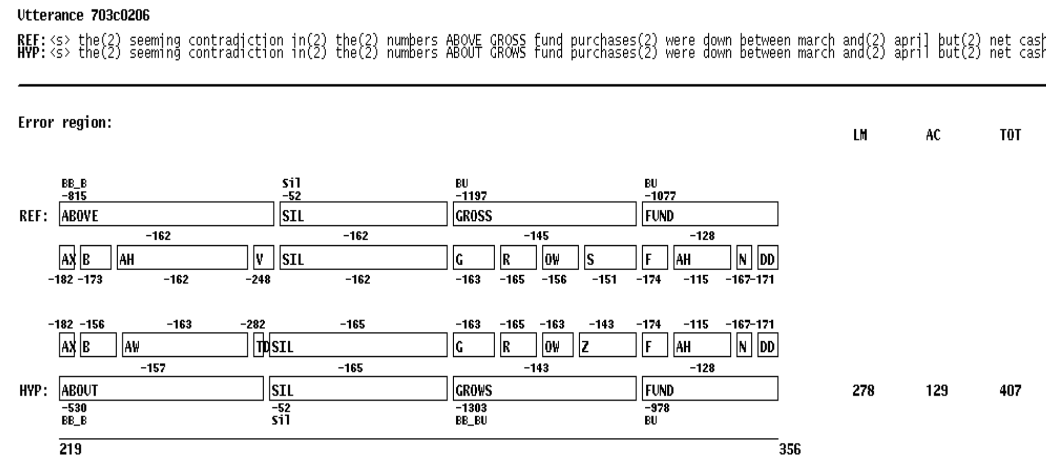
The small numbers below the REF phones and above the HYP phones in the figure are the duration-normalized acoustic phone scores. The small numbers centered below the REF word boxes and above the HYP word boxes

[3]Some recognizers do bigram and trigram search in both the forward and backward directions. The method described here assumes that the search moves forward, and that the language model employed is also a forward (or left-to-right) model. To support backward search, the only change necessary is to extend the window to the left instead of the right by the appropriate value. This is not supported in the current version of the software.



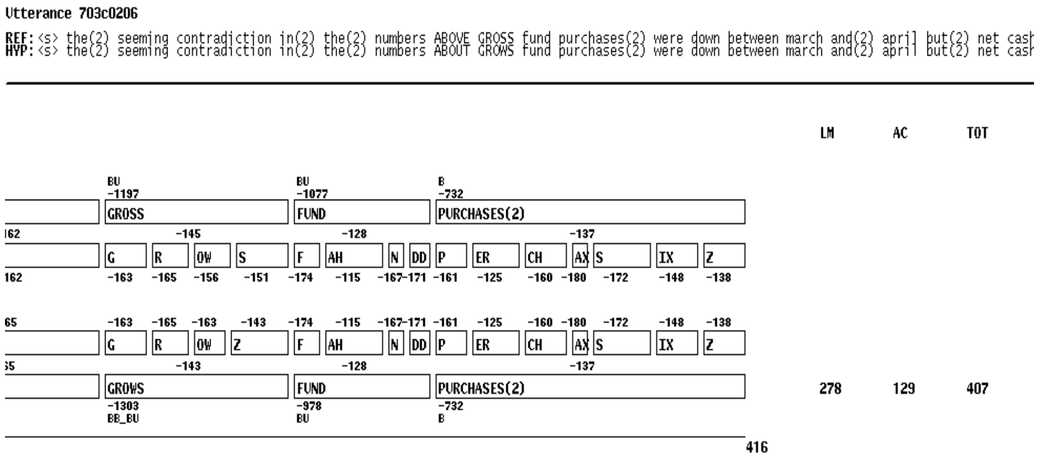
```
Utterance 703c0206

REF:<s> the(2) seeming contradiction in(2) the(2) numbers ABOVE GROSS fund purchases(2) were down between march and(2) april but(2) net cash
HYP:<s> the(2) seeming contradiction in(2) the(2) numbers ABOUT GROWS fund purchases(2) were down between march and(2) april but(2) net cash

Error region:                                                               LM      AC      TOT
```

Figure 4.3: An example utterance analyzed under the ERA technique with a frame tolerance 0 and window size 0.


Utterance 703c0206

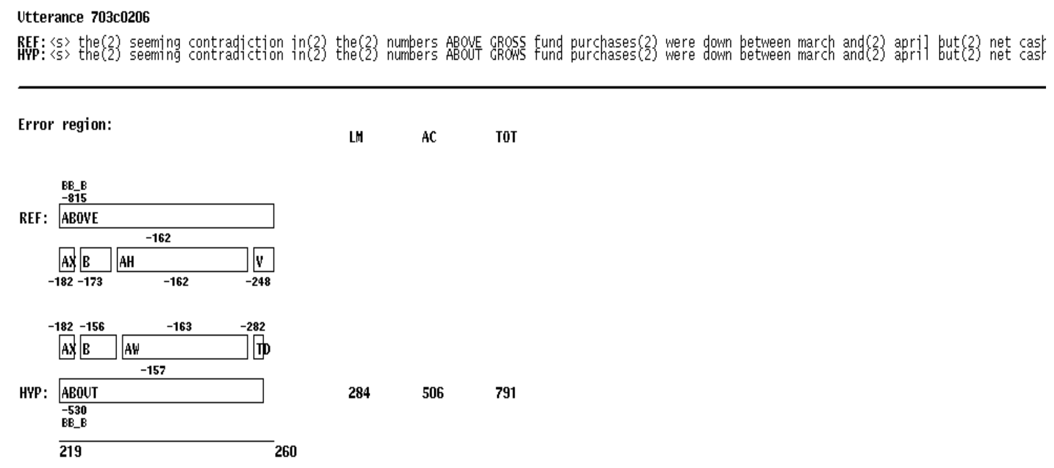
REF: <s> the(2) seeming contradiction in(2) the(2) numbers ABOVE GROSS fund purchases(2) were down between march and(2) april but(2) net cash
HYP: <s> the(2) seeming contradiction in(2) the(2) numbers ABOUT GROWS fund purchases(2) were down between march and(2) april but(2) net cash

Figure 4.4: The same utterance with frame tolerance zero 0 and window size 1.



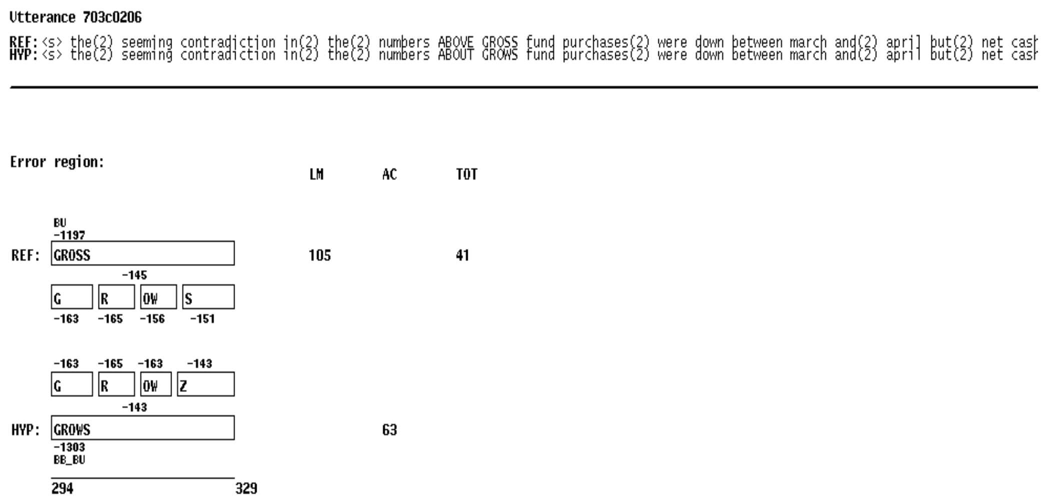
Utterance 703c0206
REF: <s> the(2) seeming contradiction in(2) the(2) numbers ABOVE GROSS fund purchases(2) were down between march and(2) april but(2) net cash
HYP: <s> the(2) seeming contradiction in(2) the(2) numbers ABOUT GROWS fund purchases(2) were down between march and(2) april but(2) net cash
LM
AC
TOT
GROSS
FUND
PURCHASES(2)
GROWS
278
129
407
416

Figure 4.5: The same utterance with frame tolerance zero 0 and window size 2.



```
Utterance 703c0206

REF: <s> the(2) seeming contradiction in(2) the(2) numbers ABOVE GROSS fund purchases(2) were down between march and(2) april but(2) net cash
HYP: <s> the(2) seeming contradiction in(2) the(2) numbers ABOUT GROWS fund purchases(2) were down between march and(2) april but(2) net cash

Error region:
                                   LM        AC        TOT
HYP: ABOUT                         284       506       791
```

Figure 4.6: The same utterance with frame tolerance zero 10 and window size 0. With these values, two error regions are found instead of one. This is the first.



Utterance 703c0206

REF:<s> the(2) seeming contradiction in(2) the(2) numbers ABOVE GROSS fund purchases(2) were down between march and(2) april but(2) net cash
HYP:<s> the(2) seeming contradiction in(2) the(2) numbers ABOUT GROWS fund purchases(2) were down between march and(2) april but(2) net cash

Error region:

Figure 4.7: The same utterance with frame tolerance zero 10 and window size 0. With these values, two error regions are found instead of one. This is the second.

are the sums of the phone scores, and are thus the word acoustic scores, again normalized by duration. [4] Note that both "ABOUT" and "GROWS" outscore their correct counterparts very slightly with respect to the acoustic models, that "AW" and "AH" seem to be virtually indistinguishable from each other acoustically in this case, and that "Z" scores far better than "S".

The small numbers at the left side of and just above the REF (below the HYP) word boxes are the language model scores associated with the transition into the word. Note that "ABOUT" scores much better than "ABOVE" in the context "THE NUMBERS", and that the HYP sequence language model score is better by 178 points. [5]

In Figure 4.4 we see the same example analyzed with the same frame tolerance but with a larger window of analysis (window size 1). This extends both the REF and HYP error region components by the word "FUND". Each of these extensions have the exact same segmentation and acoustic score. Each receive a distinct language model score, however, as the context words for the prediction of "FUND" are different. In this case, we see that the language model score difference increases in favor of the HYP, due to the higher probability $P(\text{“FUND”}|\text{“ABOUT”}, \text{“GROWS”})$.

As described in Section 2.4, under the trigram backoff language model used in these experiments, calculation of a conditional probability estimate $P(w_3|w_1, w_2)$ will fall into one of five categories:

1. *T:* The trigram (w_1, w_2, w_3) is present in the language model database.
2. *BB-B:* The bigram (w_1, w_2) is present and the bigram (w_2, w_3) is present.
3. *B:* Only the bigram (w_2, w_3) is present.
4. *BB-BU:* Only the bigram (w_1, w_2) is present.
5. *BU:* Neither (w_1, w_2) nor (w_2, w_3) is present.

[4]The visual error analysis tool used to generate these figures allows the user to select between the presentation of normalized acoustic scores and raw acoustic scores in the case of the use of frame tolerance of zero.

[5]Actually, these scores are 1000 times bigger than indicated, but have been truncated for readability. Keep in mind that, as the language model scores are based on log values computed from language model probabilities, more negative numbers correspond to worse language model scores. The closer a negative log score is to zero, the better it is in comparison with others. In the examples shown in this chapter the log base used is 1.0001.

Both the REF and HYP predictions of "FUND" in Figure 4.4 receive their language model scores from the "BU", or "backed off unigram" branch of the prediction algorithm. This means that none of the bigrams ("ABOVE", "GROSS"), ("GROSS", "FUND"), ("ABOUT", "GROWS"), or ("GROWS", "FUND") are present in the language model's database. Thus the scores are generated from a combination of information about the overall frequency of "FUND" in the language training data and the frequency with which any words follow "GROSS" and "GROWS". The point is that the language model predictions for "FUND" do not look backwards in the REF or HYP streams, and depend only on the identity of "FUND".

Thus, when the window size is extended to 2, as in Figure 4.5, there is no change in the overall analysis of the source of the error.

In Figures 4.6 and 4.7 we see the effect of increasing the frame tolerance parameter, in this case to 10. (The window size has been reset to 0.) Note that exact segmentation matches are no longer required, and that there is a SILence word in both the REF and HYP sequences between the two words in error. These facts combine to create two error regions where previously there was only one. This analysis yields the interpretation that the problem is mostly with the acoustics of "ABOUT".

Throughout this document, the settings for frame tolerance and window size should be assumed to be zero unless otherwise noted.

4.2 What can error regions tell us? (ν)

Error regions can be automatically sorted by type, based on information we receive from the recognizer and the process that creates reference alignments. A set of features that can usually be made available as input to this automatic analysis from state-of-the-art recognizers includes:

- word and phone start and end frames,
- word and phone acoustic scores,
- word language model scores, and
- word language model identifiers (tags describing the source of the language model score, for instance the branch of the Katz backoff algorithm used).

The recognizer gives us this information for the entire hypothesized (HYP) decoding. The acoustic aligner and the language model used in the recognizer give us this information for the reference (REF) sequences.

Based on this input, error regions are identified in the utterance as described in the previous section. Within each identified error region, this information is used to calculate the following:

1. *Acoustics:*
   - the total acoustic score of the hypothesized words in the error region,
   - the total acoustic score of the reference words in the error region,
   - the difference between the two, noting whether the reference or hypothesized acoustic score is better.
2. *Language model:*
   - the total language model score of all of the words in the hypothesis that fall into the error region,
   - the total language model score of all of the reference words in the error region,
   - the difference between these two, noting which is larger.
3. *Totals:*
   - the total combined acoustic and language model score for the hypothesis portion of the error region,
   - the total combined acoustic and language model score for the reference segments in the error region,
   - the difference between these two totals, noting which is larger.

As shown in Figures 4.3 through 4.5 in the previous section, these calculated quantities can be laid out visually for easy interpretation. In Figure 4.3, for example, we indicate that the hypothesis language score is better by 178 points by putting "178" on the HYP line of the display in the "LM" column. The hypothesized acoustic score is also better, by the "129" points shown in the "AC" column. Commensurately, the "TOT" figure of "308" is the sum of these two.

For any error region three of the locations REF/LM, REF/AC, REF/TOT, HYP/LM, HYP/AC, and HYP/TOT will end up containing a score difference. At least two of these score differences will be on the same line. Only one score difference will appear in each of the three columns. Thus it is possible to classify error regions as falling into one cell of a 2-by-3 matrix, as shown in Table 4.1.

To see how this works, consider the following cases. If the REF/TOT location is filled in, the utterance will be classified into one of the cells in the "REF" column, meaning that overall the REFerence score was better in the error region. For this REF/TOT error region, if both the REF/LM and REF/AC locations are filled in, then the error region will be classified as belonging to cell #3, which is in the "AC+LM" row. This means that the reason the REFerence score was better was that both the acoustic and language scores were better.

Similarly, if the REF/LM and HYP/AC locations are filled in, then the error region will be counted in cell #2, which is in the "LM" row. This means that the reason the REFerence score was better was that its language model score was enough better than the HYP's, by enough to overwhelm the superior HYP acoustic score.

The classification of error regions into the "HYP" column is analagous for those regions that have the HYP/TOT location filled in.

In short, in classifying error regions in Table 4.1, the column of the table is determined by whether the REF or HYP had the better overall score. The row is determined by which of acoustic, language, or both acoustic+language, caused the HYPothesis to be chosen over the correct answer.

Note that the cell membership of an error region does not indicate that the corresponding models are *wrong*, only that the acoustic and/or language model contribution overwhelmed the scoring process. For instance, if a HYPothesis is chosen because of an overwhelming (but error-making) contribution from the language model, it may very well be that the language model is not "wrong". Our language models are maximum likelihood models. This means that occasionally someone will utter something that is not considered likely by the language model. Such "language model surprises", or low probability events, will be hard to detect unless the acoustic models, dictionary and search strategy work extremely well to counter balance the surprise effect. An error that occurs under such circumstances can be accurately ascribed to "language score overwhelm". But it is not necessarily the case the the language model was "wrong" to be surprised by the event – the language model may in fact be doing exactly what we want it to! This point is important and will be revisited several times throughout the document.

	REF total better	HYP total better
AC bigger	(1) REF acoustics dominate HYP language model	(4) HYP acoustics dominate REF language model
LM bigger	(2) REF language model dominates HYP acoustics	(5) HYP language model dominates REF acoustics
LM bigger	(3) REF acoustics and language model both better than HYP	(6) HYP acoustics and language model both better than REF

Table 4.1: Error regions can be categorized as belonging to one of these six categories, based on the HYP and REF acoustic and language model scores.

If, as in cells 1 through 3 in Table 4.1, the reference alignment scores better in total than the hypothesis, then the recognizer's models would have made the correct prediction if given the chance. They can only have been denied the chance through some sort of *search error*.[6] For relatively "clean acoustic" tasks such as the NAB dictations, most state-of-the-art systems are tuned with various search heuristics and pruning thresholds to avoid this class of error almost entirely.

There are, however, several types of possible search errors that can be generated by the system used in this work for the NAB experiments (see Chapter 2). The first, which is not frequent, is a pruning error in the original Viterbi system. The second is a search error made in the A^* pass of the original system, usually due to the acoustic score approximation used in that pass. This is also infrequent. The most common source of search errors in this work arises as an effect of using lattices as an intermediate representation from which to generate hypotheses. The Sphinx-II lattice search process recalculated the acoustic scores of lattice word segmentations in order to produce phone segmentations. It does not allow shifts in word boundaries. Sometimes minor shifts in word boundaries would produce higher scoring word segmentations, but these are overlooked by the current system. When

[6]Many but not all of these errors are due to pruning in the Viterbi/beam search passes.

the recognizer output of this slightly suboptimal lattice search system is compared with forced alignments that are not constrained to any particular word boundaries, a number of search errors appear in the error analysis. (See the explanation of class "Miscellaneous" in the error blame assignment algorithm discussion, Section 9.3 for further details on this effect.)

If, however, as in cells 4 through 6 in Table 4.1, the hypothesis segments in an error region score better than the reference sequence, then we are faced with a *modeling error*. Let's consider each category of modeling error in turn:

1. If the HYP acoustic (HYP/AC) score is better than the REF's by enough to overwhelm the REF's better language model score, as in cell #4, then the current language model is good enough to favor the correct answer but the combined effects of the dictionary and acoustic models do not provide an appropriate model for the current acoustic situation in combination with the current language model. This could happen for any number of reasons. Some of the most common problems that lead to this effect are:[7]

   (a) speech that is not modeled well by the combination of the word reference transcript and the available dictionary pronunciations. This includes fast speech and its commensurate clitizations, phonetic deletions, and extended phonetic colorings, as well as missing alternate pronunciations and wrong pronunciations, and incorrect word transcriptions, or

   (b) the presence of confusions between acoustic models that allow data that actually represent one phone to be decoded as another with a high score.

2. If the HYP language model score is better than the REF's by enough to overwhelm the REF's correct acoustic score, as in cell #5, then the combined dictionary and acoustic models are doing the right thing but the language model is leading us astray.[8]

[7]See Chapter 9 for more detail on identifying these cases automatically.

[8]Recall that because the language models we are using are constructed to predict the likeliness of a sequence of words, the language model may not actually be *wrong* in any real sense – the REF sequence may actually simply be a low probability event!

3. If an error region is classified into cell #6, then the HYP total score is better than the REF total score because of a combined effect of the acoustic and language models.

With these definitions in mind we now move ahead to discuss errors at a lower level of detail.

4.3 What's an errorful phone prediction? (ν)

A question related to defining and using error regions is that of identifying correctly and incorrectly hypothesized phones within the HYP sequence. [9] There are two basic options for deciding whether a phone hypothesis is correct. The first [10] is to label all phones in the hypothesis portion of the error region as being incorrect, having participated either directly or indirectly in the creation of an error. This method does not prove terribly useful, as a majority of hypothesized phones in error regions in the data used in this work actually align perfectly with their reference counterparts. Thus, this first option can become a kind of blurry "guilty by association" model of phone error that captures as much information about what words are likely to be hypothesized incorrectly (via phone membership in pronunciations) as about actual phone modeling problems.

The second option for defining phone correctness is to check for actual alignment at the phone level against the REFerence sequence. In comparison with the first method, this is much more accurate.

Consider the phones in Figure 4.3. Under the first method, all the HYP phones would be labeled as incorrect. Under the second only "B", "AW", "TD", "SIL", "OW", and "Z" would be labeled as incorrect. "B","SIL", and "OW" are wrong because their segmentations do not match the reference. The others are wrong because both their identities and segmentations fail to match correctly. All other phones in the entire HYP sequence are labeled as correct, including those not in the error region.

In the example in Figure 4.3, as in general, the phones of the REFerence sequence can also be labeled as correct or incorrect. In this example, "B",

[9]Many of the analyses that follow will quote results both at the error region level and at the (lower) phone level. This is the reason this discussion appears at this location.

[10]This approach is not used in this work.

"AH", "V", "SIL", "OW", and "S" will each be labeled as incorrect because for each there does not exist a correct match in the HYPothesis sequence.

This (second) "alignment match" method will be the default method used in this work for determining REF and HYP phone correctness. It can be generalized to include a frame tolerance in the match process, exactly as with the definition of error regions.

4.4 Types of Error Regions (ν)

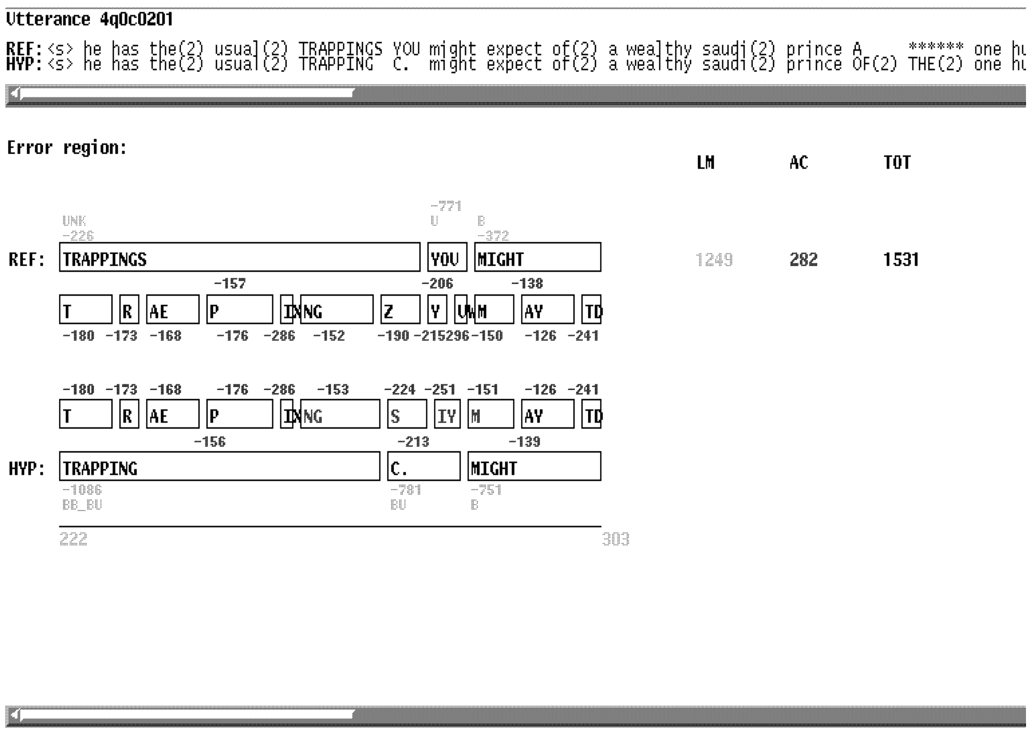
Several useful patterns emerge when looking for types of error regions created by the recognizer. These are explored next.

4.4.1 OOV Error Regions (ν)

The most obvious cause for errors is the presence of out-of-vocabulary (OOV) words in the acoustic data. These errors are detected by ERA with the help of the language model and dictionary as applied to the REFerence sequences. Any word found in the REF transcript that is not covered is marked "UNK" (for "unknown word"). Any error region that contains an OOV word is classified as having been caused by this word's (or words') presence.

Figure 4.9 shows a typical error region of this type. It is quite common to see that the recognizer responds to the presence of an OOV by recognizing (incorrectly) two shorter words that are acoustically similar to the OOV, and which together do not create a language model score that is so low as to cause the option to be pruned from the search. In this case, which is very typical, the recognizer has chosen two short words ("WILL" and "LEAD(2)") that together are similar acoustically to the spoken word, "WALEED", as its best guess. The total language model score of the sequence "WILL LEAD" is "reasonable", meaning that the overall path score generated by their presence allows the path to compete successfully with others. The combination of a cobbled-together pronunciation and the flexible backoff language model produce a viable mechanism for "modeling" (or covering) out-of-vocabulary words (OOVs).

Another example of this phenomenon can be found in Figure 4.8. In this case the plural form "TRAPPINGS" was OOV, but the singular "TRAPPING" was in the vocabulary. The short word "C." was used to cover the



```
Utterance 4q0c0201

REF:<s> he has the(2) usual(2) TRAPPINGS YOU might expect of(2) a wealthy saudi(2) prince A      ****** one hu
HYP:<s> he has the(2) usual(2) TRAPPING  C.  might expect of(2) a wealthy saudi(2) prince OF(2) THE(2) one hu

Error region:
                                                      LM      AC      TOT
REF:  TRAPPINGS  YOU  MIGHT                           1249    282     1531
HYP:  TRAPPING   C.   MIGHT
```

Figure 4.8: An error region caused by an out-of-vocabulary (OOV) word in the utterance. From the NAB development test set, with phone segmentations and scores included.

final “S” sound in the OOV word, and the backoff mechanism was used to find the “reasonable” score for the sequence “TRAPPING C.” in the hypothesis.

For additional details on this type of behavior, and some discussion of what properties we can detect in a more automated fashion, see both Section 6.9 and Chapter9.

Throughout most of this work OOV error regions are treated entirely separately from other types of error regions. This is because:

1. for the data and recognizer studied in this work (see Chapter 9) the error regions caused by OOV words have a very clean separation from those that arise due to other problems,

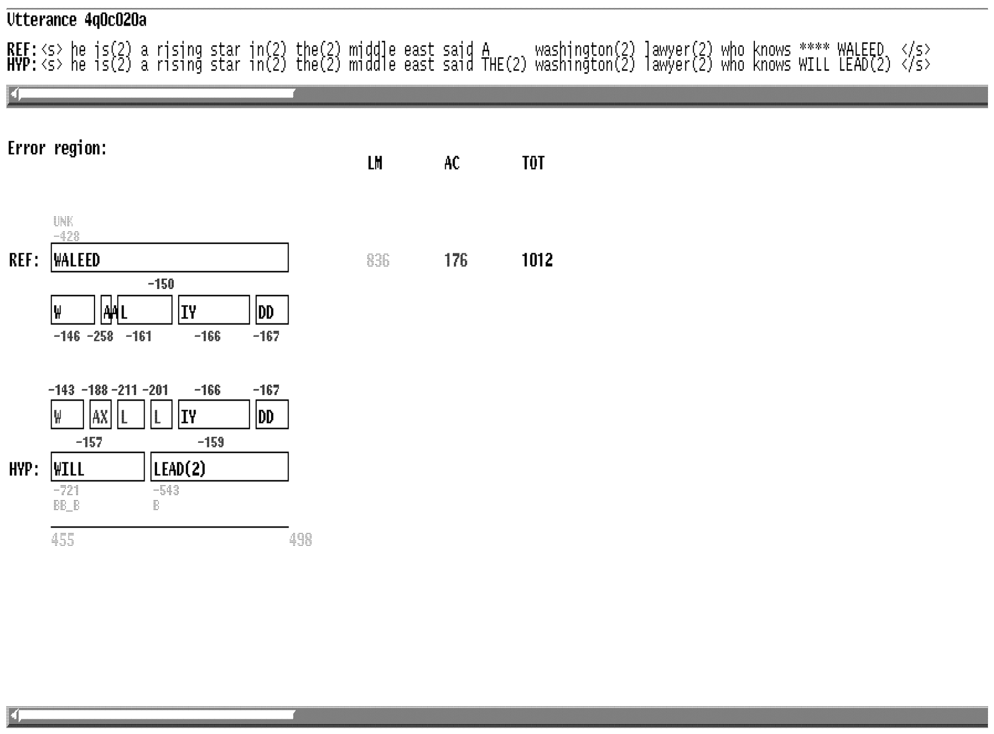
2. the recognizer behaviors observed in dealing with OOV words are distinctive and constitute a nicely separable prediction problem (see Chapter 6, and

3. the reliable prediction of the location of OOV words as a special case of error prediction is a highly valuable capability, especially in applications in which it is not possible to use extremely large vocabularies.

4.4.2 When the Language Model Leads Us Astray (ν)

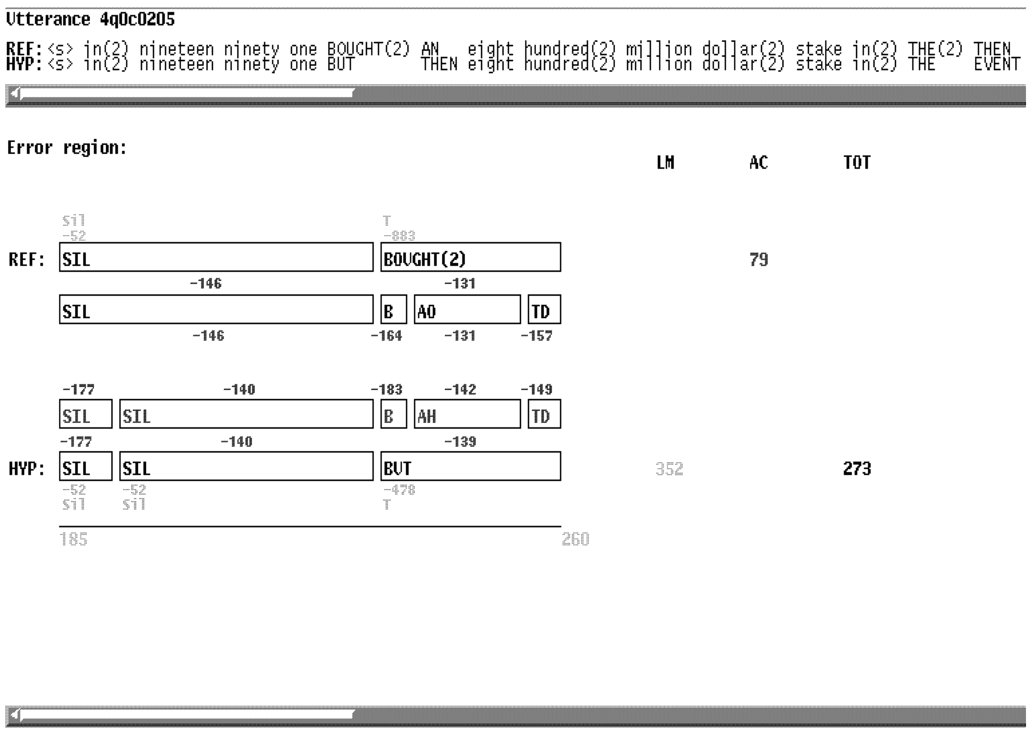
As described in Section 4.2, if the reference segments in an error region have a better acoustic score than the hypothesis segments, and the language model score for the incorrectly produced sequence is better than for the reference sequence, then the language model can be thought of as having led us astray. Examples of this type of error region can be found in Figures 4.10 through 4.12. Recall that in general the language model may not actually be “wrong” to prefer the incorrectly hypothesized sequence – the REF sequence may simply be a low probability event! What is missing is the ability to determine that in these cases the acoustics are actually doing the right thing and should be relied upon. That is, what is needed in these cases is not necessarily a change in the language model score produced, but rather an adjustment to the relative contribution of the acoustic and language models, possibly through dynamic adjustment of the language weight LW, as in [41].

For more details on this type of behavior, see Chapter 9.



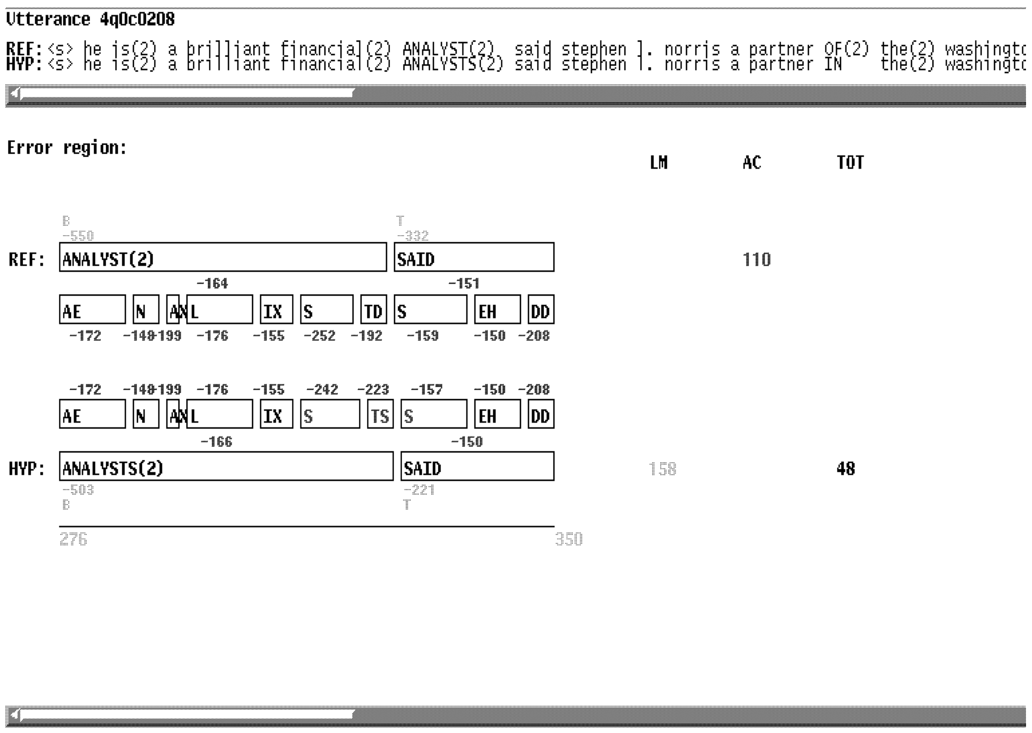
```
Utterance 4q0c020a

REF:<s> he is(2) a rising star in(2) the(2) middle east said A      washington(2) lawyer(2) who knows **** WALEED  </s>
HYP:<s> he is(2) a rising star in(2) the(2) middle east said THE(2) washington(2) lawyer(2) who knows WILL LEAD(2) </s>
```

Figure 4.9: Another OOV error region from the NAB development set. Phone segmentations and scores included.



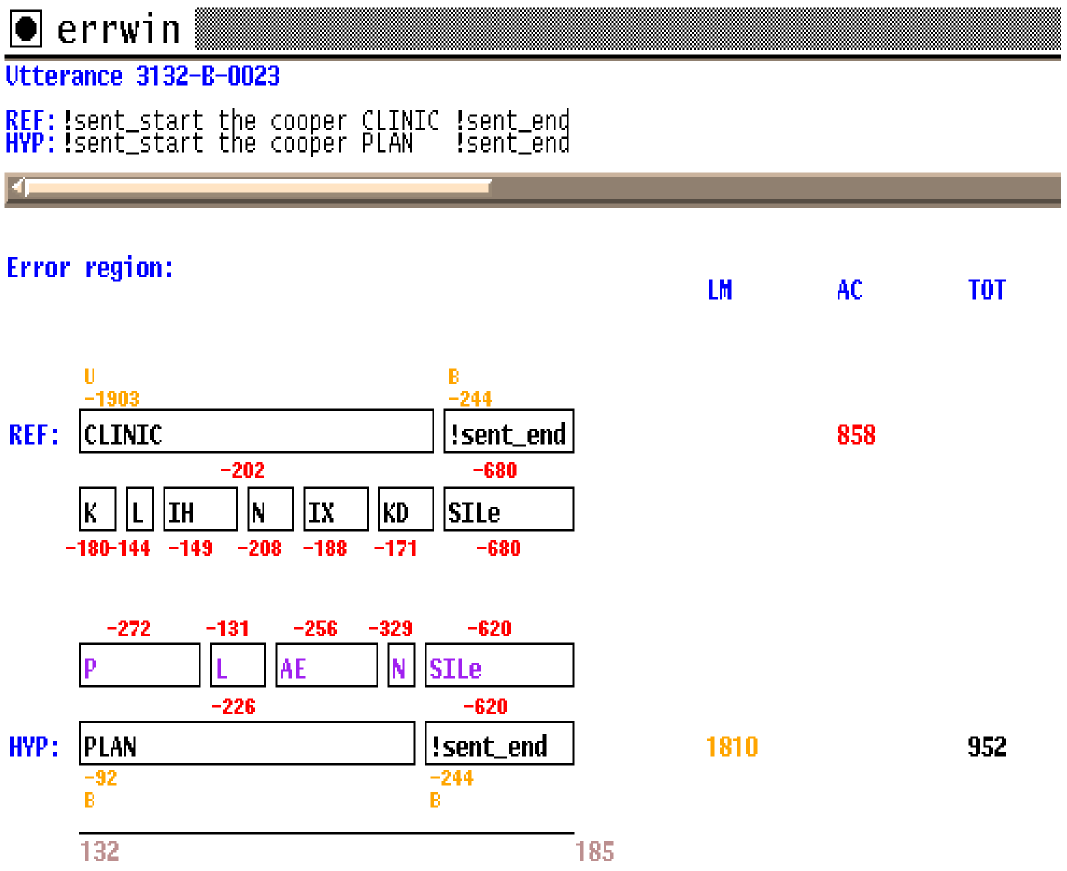
```
Utterance 4q0c0205

REF:<s> in(2) nineteen ninety one BOUGHT(2) AN   eight hundred(2) million dollar(2) stake in(2) THE(2) THEN
HYP:<s> in(2) nineteen ninety one BUT          THEN eight hundred(2) million dollar(2) stake in(2) THE    EVENT
```

Figure 4.10: An error region caused by an over-enthusiastic trigram language model. From the NAB development test set, with phone segmentations and scores included.



```
Utterance 4q0c0208

REF:<s> he is(2) a brilliant financial(2) ANALYST(2)  said stephen l. norris a partner OF(2) the(2) washingtc
HYP:<s> he is(2) a brilliant financial(2) ANALYSTS(2) said stephen l. norris a partner IN    the(2) washingtc

Error region:
                                                      LM        AC        TOT

REF: ANALYST(2)   SAID                                          110

HYP: ANALYSTS(2)  SAID                                158                 48
```

Figure 4.11: Another error region caused by an over-enthusiastic trigram language model. From the NAB development test set, with phone segmentations and scores included.


errwin

Utterance 3132-B-0023

```
REF: !sent_start the cooper CLINIC !sent_end
HYP: !sent_start the cooper PLAN   !sent_end
```

Error region:

Figure 4.12: An example from SWITCHBOARD in which the language model overwhelms correct acoustic scoring.

4.4.3 When the Acoustic Score Leads Us Astray (ν)

As the complement of what we saw in the previous section, it is also possible that the total acoustic score in an error region may overwhelm an otherwise correct decision on the part of the language model. As described in Section 4.2, if the reference segments in an error region have a better language score than the hypothesized segments, and the acoustic model score for the incorrectly produced sequence is better than for the reference sequence, then the acoustic models have led us astray. An example of this type of error region can be found in Figures 4.13 through 4.15. There are many reasons why the recognizer may prefer the acoustics of an incorrect hypothesis. One possibility is that the acoustics of the signal may not be anything like the speech models the recognizer is using – there may be loud noises overwhelming any speech signals present, for instance. Another possibility is that the speaker used an unmodeled pronunciation of a word or phrase. Another possible problem might be that two or more of the phone acoustic models are not trained well enough to avoid confusions between themselves. Or finally, the word reference transcript might simply be wrong.

What is missing in this case might be the ability to determine that in these cases the acoustics are not doing the right thing and that the language model should be relied upon instead. In the case of confusions between acoustic models, however, corrective training of the models is probably in order, as in [34].

For more detail on how we can sort through these different cases in a more automated fashion, see Chapter 9.

4.5 Summary

We have seen that by looking closely at what is going on frame by frame in the acoustic and language scoring components of the recognizer that we can start to understand why particular errors are made. This analysis relies on having a reasonable definition of what the "correct" sequence of states "should" have been as a point of comparison with the recognizer's best guess. Examples of how one or both of the language and acoustic scores can overwhelm an otherwise correct decision were given. We also saw examples of how the recognizer tends to handle out-of-vocabulary words. Further details on this approach are given in Chapter 9.

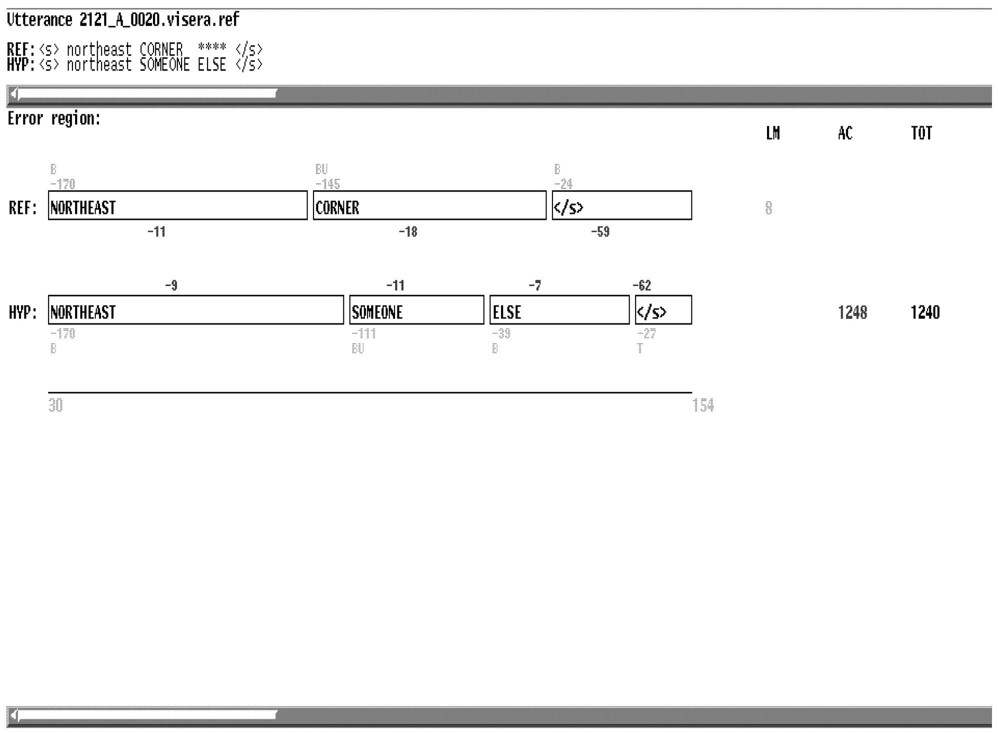
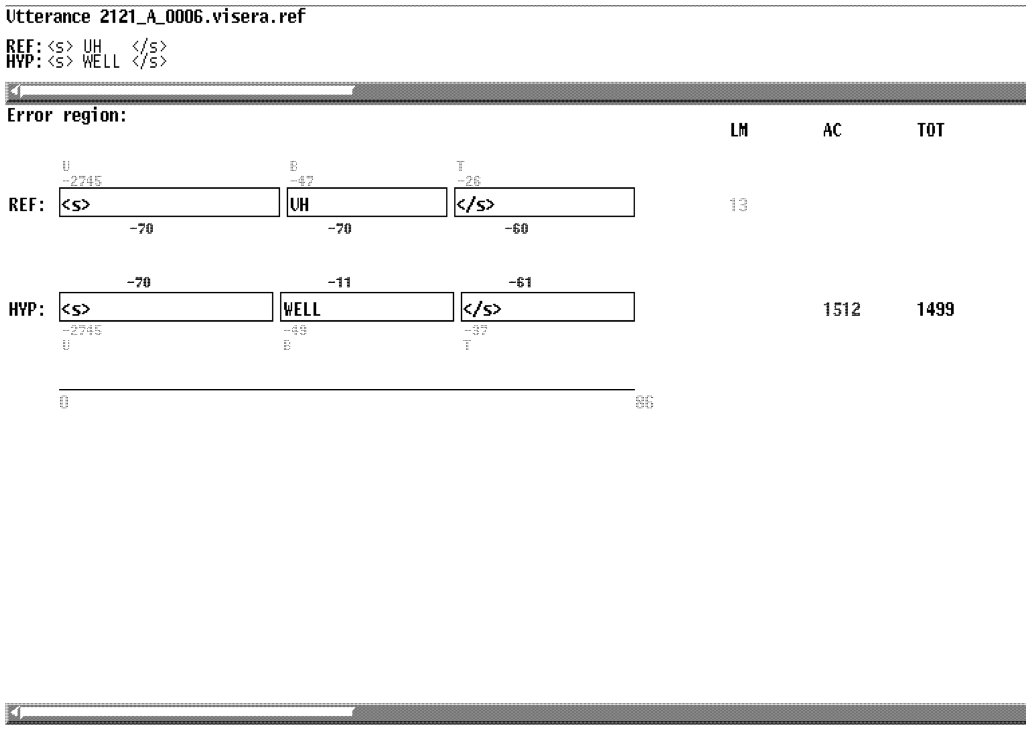
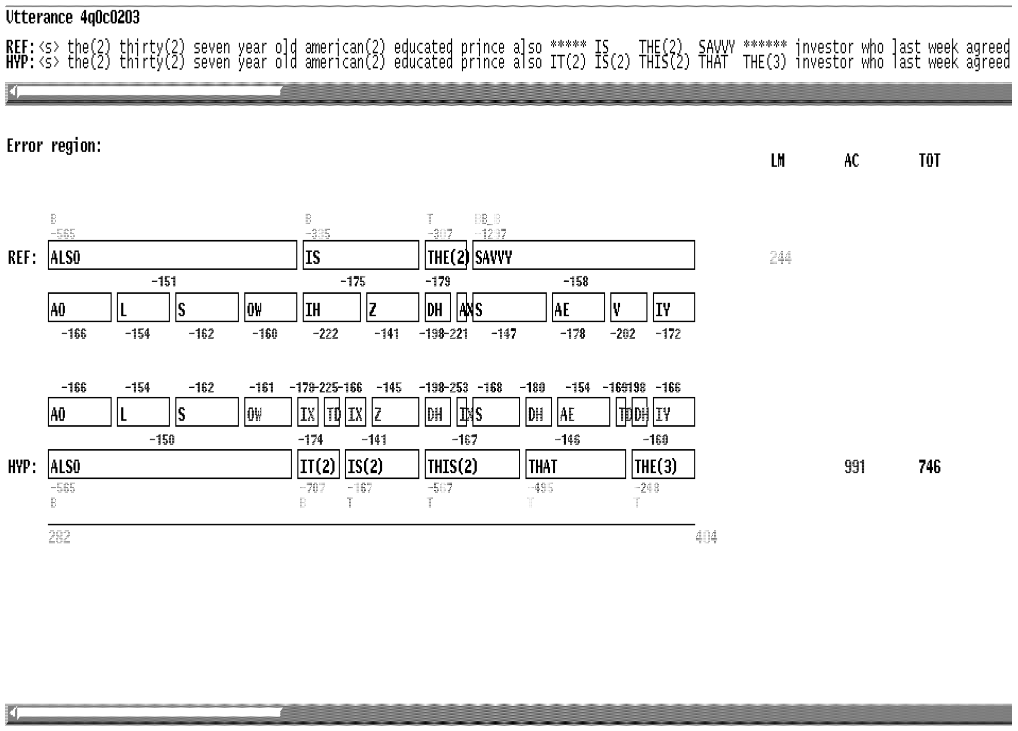

```
Utterance 2121_A_0020.visera.ref

REF:<s> northeast CORNER  **** </s>
HYP:<s> northeast SOMEONE ELSE </s>
```

Figure 4.13: Example in which the HYPothesized acoustics score better than the REFerence acoustics. From Switchboard development data.



```
Utterance 2121_A_0006.visera.ref

REF:<s> UH   </s>
HYP:<s> WELL </s>

Error region:
                                                        LM      AC      TOT

        U                  B            T
        -2745              -47          -26
REF:   <s>                 UH           </s>            13
             -70                -70           -60

             -70                -11           -61
HYP:   <s>                 WELL         </s>                    1512    1499
        -2745              -49          -37
        U                  B            T

       0                                          86
```

Figure 4.14: Another Switchboard error region produced by an overwhelming acoustic score.



```
Utterance 4q0c0203

REF:<s> the(2) thirty(2) seven year old american(2) educated prince also ***** IS    THE(2)  SAVVY ****** investor who last week agreed
HYP:<s> the(2) thirty(2) seven year old american(2) educated prince also IT(2) IS(2) THIS(2) THAT  THE(3) investor who last week agreed

Error region:                                                              LM      AC      TOT
```

REF:
ALSO
IS
THE(2)
SAVVY
244
AO L S OW IH Z DH AX S AE V IY
HYP:
ALSO
IT(2)
IS(2)
THIS(2)
THAT
THE(3)
AO L S OW IX TD IX Z DH IX S DH AE TD DH IY
991
746
282
404

Figure 4.15: Another acoustic score overwhelms correct language scoring. From the NAB development data, with phone segmentations and acoustic scores shown in addition to word-level information.

Chapter 5

A Baseline Confidence Annotator

One of the main goals of this work is to take advantage of information fed back to the recognizer about its successes and failures at guessing word sequences. This chapter describes an approach that uses feedback data to create a probabilistic error predictor. Error predictors (interchangeably called "confidence annotators") are used to identify regions of high and low confidence in the recognizer's best guess. I have extended this definition to additionally include the identification of specific types of errors, especially those due to the presence of OOV words in the input.

Recognizer output labeled with confidence annotations is richer than "flat guesses", and can be used in a variety of applications:

1. When producing speech-to-text transcriptions, having a confidence measure reduces the amount of hand error checking necessary to produce a final text.

2. When automatically processing or classifying recorded data, as in multimedia storage and retrieval applications [21], having a measure with which to weight likely "hits" based on recognizer performance reduces error.

3. In human/computer dialogue and automatic translation systems regions of low confidence can be made the focus of clarification actions.

4. In multi-pass recognition systems that support tradeoffs between speed and accuracy, confidence annotation of early, inexpensive phases can

be used to focus computational resources on the more difficult sections of the input.

5. Confidence annotation can be used to dynamically adjust the local contribution of acoustic and language models [41].

6. Unsupervised acoustic adaptation can be improved by using a confidence measure to filter adaptation data. (See Section 8.3.3.)

7. When automatically processing large amounts of speech, confidence annotation can be used to identify portions of a database for which the recognizer is not working well. (See Section 8.3.3.)

Given this motivation, the task remains to answer the following design questions for confidence annotators:

1. At what level of abstraction should we annotate confidence?

2. What is the right way to define what is an error and what is not when generating feedback data?

3. What mechanism should we use to create confidence annotations?

   - What predictor variables are useful? How useful?
   - How should we combine evidence from the various predictors to create a confidence annotation?[1]

4. How do we measure the goodness of our confidence predictions?

The four questions above outline the general design space of any confidence annotation approach. This chapter describes the answers to these questions as implemented for two simple confidence annotators built for use as baseline performance systems for this thesis.

The following sections address each of the four design questions in turn as they apply to the baseline system. Later chapters address the questions for more sophisticated, powerful and detailed confidence annotation schemes, together with performance improvement results.

[1]As described in the next chapter, predictor variables typically combine in a non-linear fashion. Most predictor variables are clearly not independent of at least some others.

5.1 What Level of Abstraction? (ν)

Because all previous related work in confidence annotation [12] [6] [51] [20] [5] [41] [8] [10] has been done at the word level, and because it is very simple to implement, the baseline metric is defined and tested on a word-by-word basis. All previous related work modeled only one error class, *incorrect*. I extend the technique by tagging words as either *correct* or as belonging to one of several error categories. The baseline system reported in this chapter tags words with probabilities of being either correct (*correct*), incorrect due to the presence of an out-of-vocabulary word in the input (*incorrect/oov*), or incorrect for other miscellaneous reasons (*incorrect/other*). This means that our confidence annotations (error predictions) consist of labeling *words* with probabilities[2] that correspond to the likeliness that the word in question belongs to each of the classes under consideration.

Thus each possible class label for a word is given a possible score between zero and one. The set of class scores for a word are constrained to sum to one.

Chapters 6 and 8 explore how best to do word-level confidence annotation on two continuous speaker-independent tasks: large-vocabulary dictation and conversational telephone speech. In Chapter 7 we modify the level of abstraction at which labels are applied by moving down to the phone level of abstraction. Another contrast arises in Chapter 9, in which the experiments rely on an abstraction level above the word – the error region from Chapter 4. (This work is not on confidence annotation, rather its cousin, error blame assignment.)

5.2 What is and is not a correct word? (ν)

In most of the confidence annotators described in this work we create feedback data at the word level in a slightly different fashion than is generally the case in the research community at large. It is common elsewhere to create feedback data using only lexical information from the word-level transcripts. Often the REFerence transcript is compared with the recognizer's best HY-

[2]The scores assigned to words behave mathematically like probabilities. In most cases the scores actually do represent *a posteriori* probabilities of labels. In some cases, however the scores cannot be strictly considered to be probabilities, even though they behave mathematically as such.

Pothesis without regard to frame-level segmentation information. This is an underestimating feedback mechanism – as we saw in Chapter 4, several kinds of important errors can be missed when using this technique.

This underestimation problem alone is enough to motivate a stronger error-labeling scheme, but in addition there is another reason to be more detailed in this activity. In particular, later extensions of this scheme allow us to predict error classes at the phone level of abstraction. (As described in detail in Chapter 7 this makes it possible to measure the reliability of the acoustic models at any point in time.) Because most phone errors do not occur in any clear relation to lexical boundaries that appear in word alignments it is very important to generate phone feedback data at the frame level of abstraction. In order to provide a general framework at the word level that is compatible with the extensions that follow this baseline discussion, the following method for word error labeling is used instead of the more common method:

1. The segmented HYPothesized word is compared with all REFerence aligned segments.

2. If a lexical match is found then the start frames and end frames of the HYP and REF segments in question are compared.

3. If both the start frame and end frame of the HYP word are within some frame tolerance of the REF segment then the match is considered to be correct.

A frame tolerance of 2 frames will be used in experiments involving the NAB data and the Sphinx-II system. A frame tolerance of 3 frames will be used in experiments involving the telephone speech data and the Janus system.

By comparison with the tags created by word-level alignment [3] this is in some cases an overestimating method, as illustrated in Figure 5.1. In this example the words INVESTMENTS, IN(2), THE, STRATEGIC, NOT, TO, START,... will all be tagged as incorrect. It is arguable that INVESTMENTS and STRATEGIC might possibly be better marked as correct, since they have exactly correct phoneme sequences and only minorly incorrect segmentations of boundary phones. Both words have HYPothesized end

[3]which, you will recall, is the basis of the WER calculation

frames that exceed the frame tolerance, and thus are not marked as correct. When a word is not marked as correct, we then label it as either belonging to class *incorrect/other* or class *incorrect/oov*. We compare the time-aligned HYPothesized word with the REFerence aligned words. If there is an OOV REFerence word within some number of frames, the word is labeled *incorrect/oov*. Otherwise it is labeled *incorrect/other*. This additional frame tolerance, used only for OOV labeling, is ten frames in both sets of experiments. The use of this large tolerance means that errors created just to the left and/or right of the location of an OOV word will be tagged as having been due to the OOV word.

5.3 How do we Make Confidence Annotations? (ν)

The basic idea in building a confidence annotator is to create a system that maps recognizer-internal descriptions to externally defined error conditions. Our externally defined error conditions are labels of the three word classes

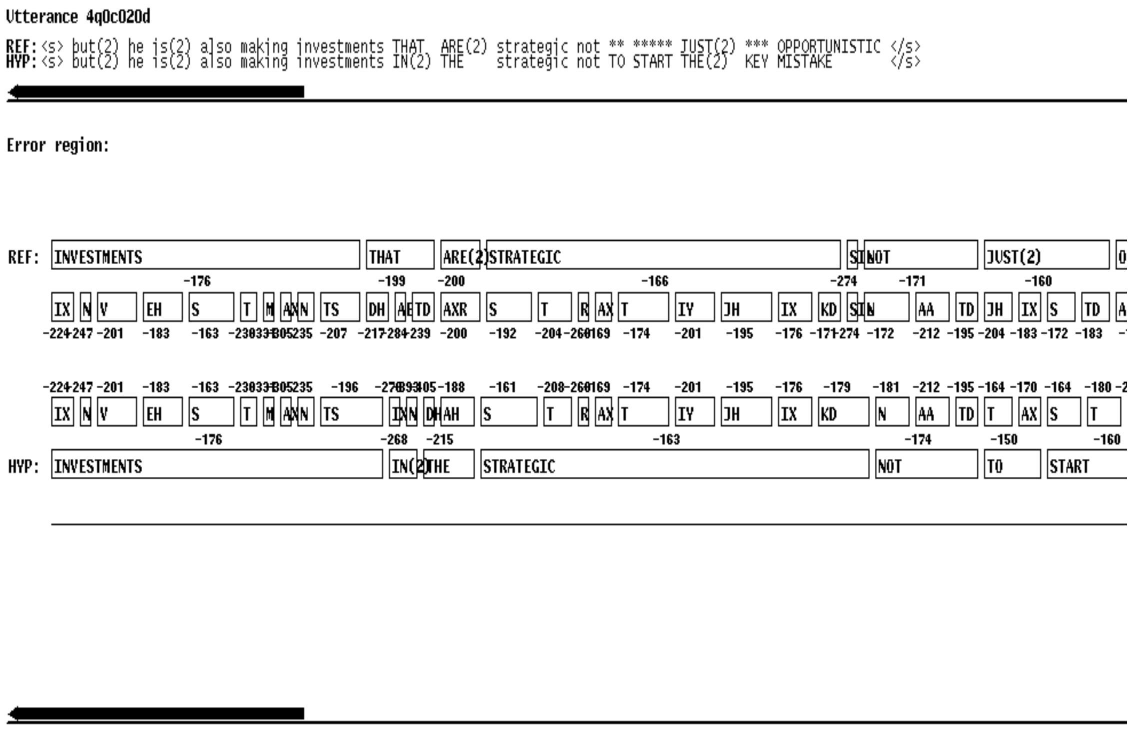
1. *correct*: the HYPothesized word matched the word found with a similar segmentation in the REFerence alignment,

2. *incorrect/oov*: the HYPothesized word was not correct, and was guessed by the recognizer in an attempt to "cover" a word in the REFerence transcript that was out-of-vocabulary (OOV), and

3. *incorrect/other*: the HYPothesized word did not match a word with a similar segmentation in the REFerence alignment, but the error was not due to an OOV word in the REFerence.

The special treatment of errors in class *incorrect/oov* is made for several reasons. First, error analysis of the style described in Chapter 4 indicated that the recognizer's response to errors due to the presence of OOVs was somewhat distinct and predictable compared to other cases. Second, the identification of the location of OOV words is in itself a useful function. Finally, it was clear that there were a fixed number of errors due to OOVs in the test sets used for the experiments.

In the ideal we would treat each distinct source of error as a separate class, predicting each separately. The limited amount of training data and



```
Utterance 4q0c020d

REF:<s> but(2) he is(2) also making investments THAT  ARE(2) strategic not ** ***** JUST(2) *** OPPORTUNISTIC </s>
HYP:<s> but(2) he is(2) also making investments IN(2) THE    strategic not TO START THE(2)  KEY MISTAKE       </s>

Error region:
```

REF:
INVESTMENTS
THAT
ARE(2)
STRATEGIC
SIL
NOT
JUST(2)
IX N V EH S T M AX N TS DH AE TD AXR S T R AX T IY JH IX KD SIL N AA TD JH IX S TD
IX N V EH S T M AX N TS IX N DH AH S T R AX T IY JH IX KD N AA TD T AX S T
HYP:
INVESTMENTS
IN(2)
THE
STRATEGIC
NOT
TO
START

Figure 5.1: The ERA-based method of tagging words as correct or incorrect can be over-aggressive sometimes. All the words in this error region will be tagged as incorrect.

the lack of complete understanding of the source of errors prevents this at present.

We represent the recognizer's internal configuration over time by using a variety of predictor variables. Most of these variables are attributes of the HYPothesized words, or descriptions about how the HYPothesized words related to other recognizer components such as N-best lists.

There are two basic types of these variables: *continuous variables* and *categorical variables*.

Continuous variables express values that can vary numerically over some (possibly very large) range, and can take on integer or real-numbered values. For instance, the observed duration of a particular triphone seen in either the acoustic training data or the error training data is expressed as a positive integer and will be treated as a continuous variable. Another example of a continuous variable is the normalized acoustic score of a word. It will typically be a floating point number that scales the ratio between the "raw" acoustic score of a word and the best possible acoustic score for the word's segmentation.

Categorical variables are labels that describe different levels of values contained in the data. For example, the source of a language model score from a Katz-style trigram model will take on one of the five values described in Section 2.4.

Identifying and evaluating the usefulness of good error predictor variables for the types of error conditions being modeled is really the key to building good confidence annotators.

In this section we will build two baseline systems, comparing each against the other and illustrating the basic process of constructing a confidence annotator.

The first baseline system, called "BASE1:percPhAll" throughout this document, compares the HYPothesized word sequence with a phone-only decoding that is done in parallel. When the phone sequence in the HYPothesized word sequence is different from that in the phone-only decoding, there may be an error in what was hypothesized by the recognizer.[4] The predictor variable is:

[4]The potential success of this approach depends in part on the quality of the phone-only decoding, which, as described in Section 2.8, is only about 72% accurate. We ignore this problem in the baseline.

- *percPhAll*: The phone-only decoding is mapped down to the frame level, as are the constituent phones of the HYPothesized words. For each word the number of correct matches at the frame level is counted, and normalized by the number of frames in the word.

The second baseline system, called "BASE2:Combined+Duration", does not rely on any side computations. Rather, it is intended to represent the ability of the recognizer to predict errors based on roughly the same information it has available for decision-making during its normal computation. Thus the predictor variables used in this baseline are:

- *Combined*: Each word output by the recognizer has an associated acoustic score that was used to compute path scores during the search. Each word also has a language model score. The two are combined using a globally fixed language weight to create the total combined word score.

- *Duration*: While the recognizer does not explicitly reason about durations, it does make a choice at most frames about whether to leave/enter a new word. Thus we include the actual duration of each word in frames in the list of basic (and therefore baseline) sources of information to be used in error prediction.

Example values for predictor and response variables used in the baseline system appear in Figure 5.2.

Later chapters describe extensive additions to this list of predictor variables. These variables will be compared to any similar approaches taken in related work, where applicable.

How to combine predictor variables to create annotations? Given the predictor and response data described in the previous section, how do we combine what we have to create confidence annotations on independent test data of the same form?

Any mechanism that can relate predictor variables to error category response variables in a probabilistic fashion can be used to create a confidence annotator. One approach that is very straightforward is the use of classification and regression trees (CART), or "decision trees".

(Most of the experiments in this work rely on decision trees. However, in Section 8.3.2 this method is compared directly with three other methods:

```
class           combined   duration   percPhAll
correct         -2804029     15         0.75
correct         -12151766    69         0.88
incorrect/other -3664532     19         0.44
incorrect/other -7526094     49         0.35
correct         -5000787     27         0.65
correct         -5017272     33         0.80
incorrect/oov   -10646200    57         0.55
correct         -6785119     42         0.70
.
.
.
```

Figure 5.2: Example predictor and response data for the baseline confidence annotator (error predictor). Note that the response variable "class" can take on one of the three values *correct*, *incorrect/oov*, *incorrect/other*. The predictor variables "combined" and "duration" can take on a wide range of integer values. The variable "percPhAll" can take on real values between zero and one.

generalized linear models, generalized additive models, and neural networks. See that discussion for a detailed discussion.)

Decision trees are trees whose nodes contain questions that can be applied to an input data element. In order to create a prediction for a given data element, we place the datum at the root node of a tree. It is then passed down a specific path through the tree to a leaf node. The path taken depends on answers to the questions found in the nodes encountered along the way. The leaf nodes (and in some cases the nodes traversed in passing through the tree) contain the probabilistic information used to create the confidence annotation for the data element in question. In this section I describe how decision trees are constructed and used in the baseline confidence annotation system.

The baseline system for this work uses decision trees trained on data from the development test sets described in Section 3.1.2. The test data is the independent evaluation test data described in 3.1.3.

The response variable used in these experiments is selected from the 3-member set $correct, incorrect/other, incorrect/oov$. This is a categorical variable, thus the decision tree produced is a categorization tree (as opposed to a regression tree).

The development data is used to construct and tune a classification tree. Each test element from the independent test data is then passed through the tree. All of the development data and all of the test data are described by the predictor features described in Figure 5.2. As a result of being passed through the decision tree, each test data element is labeled with a probability of belonging to each of the three output classes ($correct$, $incorrect/oov$ or $incorrect/other$). Each test element will receive three probability labels which sum to one. (It is possible to interpret the output probabilities of each word as predicting a one-class answer corresponding to the label that receives the highest probability score, but ties are actually possible and our goodness measures do not always require such definite decisions.)

Constructing tuned categorization trees In this section we will look at how decision trees are grown, pruned, cross-validated, and used.

Trees are grown using a binary recursive partitioning algorithm. The algorithm (explained in detail in [7]), tries to partition the training examples it works with into homogeneous groups, which correspond to regions in the space of variable values. The algorithm tries to group all training examples that share a particular value on some predictor variable y in such a way that the distribution of y in the group does not depend on the values of the other predictor values found in the set.

First all of the observed values for each training example are collected and rank-ordered. This defines (in the abstract) a set of possible *cutpoints*, which are binary decision points that create partitions. The decision tree, which is binary, is grown by searching through all cutpoints for all predictor variables at each node, selecting the next *split* from among the available cutpoints using the evaluation function described next. The goal is to make the best possible split of the training examples in the node into two children nodes.

A split is chosen when it reduces the *deviance* of the tree maximally. For categorical variables, the deviance for a single training example is a function of the log-likelihood,

$$D(\mu_i; y_i) = -2\sum_{k=i}^{K} y_{ik} log(p_{ik})$$

where i represents a single training example, y_i is the value of the response variable for i, p_{ik} is the probability of seeing the value k at instance i, and k varies over all of the K or possible categorical values.

For continuous variables, the deviance of an observation is defined as

$$D(\mu_i; y_i) = (y_i - \mu_i)^2$$

At any given node, the mean μ_i is constant for all observations in the node. The probability of membership in any given class (in our case *correct*, *incorrect/other*, or *incorrect/oov*) is given by the node proportions.

The deviance of a node is defined as the sum of the deviances of all observations in the node. The deviance will thus be zero if all of the training examples in the node are identical, and increases with decreasing homogeneity of the node.

Each successive split is made with the local goal of reducing the overall tree's deviance as much as possible. Splitting terminates when the node deviance is less than 1% of the overall tree deviance, or when there are fewer than five training instances in a given node.

This splitting algorithm produces vastly overtrained trees which need to be pruned back and smoothed. This is accomplished under a ten-way "jacknifing" [30] cross-validation (rotating through 9 parts train,1 part test)

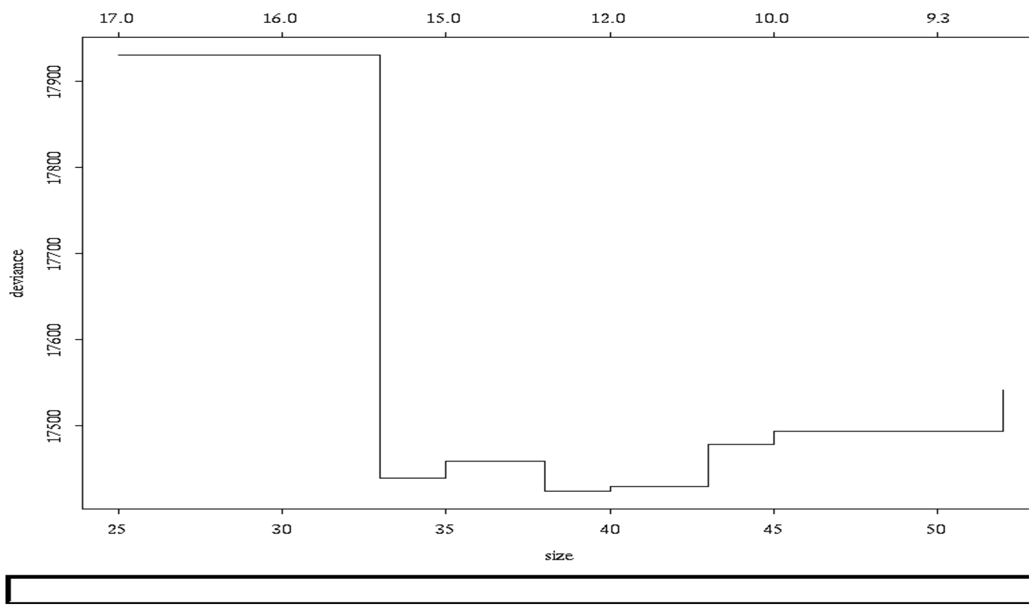


Figure 5.3: Results of cross-validation runs on the categorization tree built for the BASE2:Combined+Duration baseline confidence annotator. The lower x-axis is tree size counted by number of nodes. The upper x-axis is value of k, the pruning parameter. The y-axis is the average tree deviance under a 10-way cross-validation. The tree under analysis has 243 nodes when built using the reported splitting termination criteria.

which uses the following function to recursively snip off the least important splits. The pruning cost function is:

$$D_k(T^{'}) = D(T^{'}) + k \cdot \text{size}(T)$$

where $D(T^{'})$ is the deviance of the subtree $T^{'}$, size$(T^{'})$ is the number of terminal nodes of $T^{'}$, and k is the cost-complexity (pruning) parameter.

Running this pruning mechanism at a variety of pruning levels (values of k) creates a curve which allows us to select the optimal value according to the cross-validation. Figure 5.3 displays this curve as calculated for the BASE2:Combined+Duration system. In this case the optimal value for minimum tree divergence of this training set under cross-validation is somewhere near $k = 13$. The tree pruned with the value $k = 13$ has 38 terminal nodes

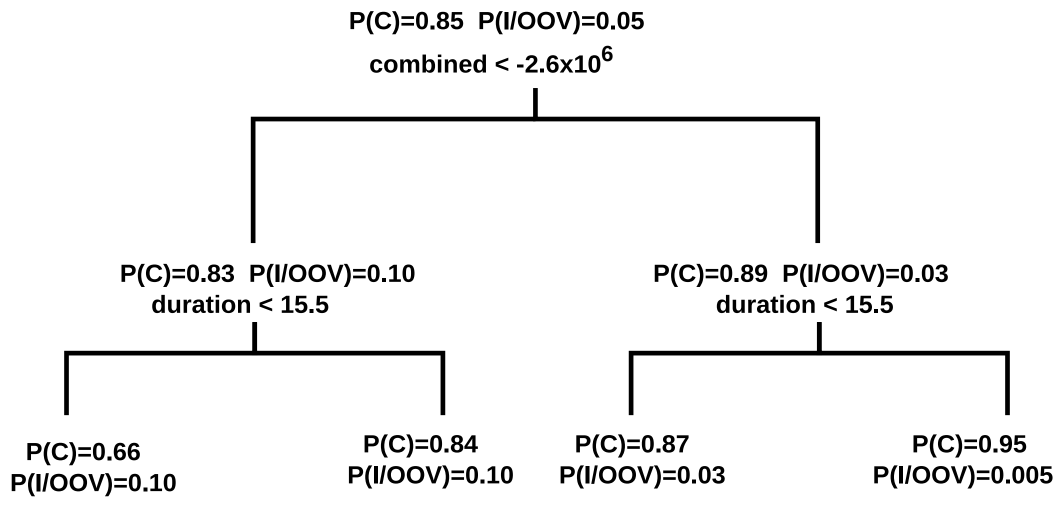


Figure 5.4: Top branches of the female baseline predictor tree.

which will be used for prediction. A graphic depiction of the top of this tree can be found in Figure 5.4.

Once this value has been selected, the appropriate tree (the one pruned using the optimal value of k) is constructed and used in labeling independent test data.

5.4 How to Measure Goodness of Confidence Annotations? (ν)

Just as there are many possible ways to create confidence annotations, there are also many ways to calculate how well a particular confidence annotation scheme performs. For any approach selected, however, it will be the case that the test data will be aligned and labeled with "true" labels exactly as the training data was labeled. The performance of a given confidence annotator will be compared against these "true" labels.

In the case of the baseline system reported here, this means that each word in the test data is marked as being either *correct*, *incorrect/oov*, or *incorrect/other*. This means that one of the three output classes has a "true" probability of 1.0 and the other two have "true" probabilities of 0.0. In order to calculate how well a particular confidence annotator performs we must compare the true labels associated with the test data to the labels assigned by the confidence annotator.

For the purposes of measuring the performance of the systems reported in this thesis I use both a general-purpose entropy-based measure and application-specific measures. The entropy-based measure, which is theoretically sound and well-motivated, can be used in any situation to evaluate competing annotation schemes in the same task, and sometimes to compare the difficulty of confidence annotation across tasks. Application-specific measures are useful in determining whether a particular confidence annotator will actually provide added value in some specific context.

5.4.1 A Metric Based on Cross-Entropy (ν)

We are interested in comparing the cross-entropy of a stream of confidence annotations with the fundamental cross-entropy of the non-annotated output of the speech recognizer. Cross-entropy is a measure that compares two distributions. In our case, the cross-entropy is measured between some set of confidence annotations and the actual class labels of the stream. (For a detailed discussion of cross-entropy and the related concept of perplexity, see [53].)

The cross-entropy of the non-annotated output of the speech recognizer can also be called the "self-entropy" of the recognizer output stream. This self-entropy measures the amount of information in the *a priori* confidence annotations that correspond to guessing the actual experience rates of each class label in the recognizer output stream. For example, in our three-class labeling example the class *correct* occurs about 83% of the time, *incorrect/other* about 12%, and *incorrect/oov* about 5%. Thus to measure the self-entropy of the recognizer output we use a default labeling of $[0.83, 0.12, 0.05]$ on each word output by the recognizer. [5] Any improvement over this basic annotation, made by a better alternative confidence annotator, will result in reduced cross-entropy.

The calculation of the cross-entropy of a labeled stream is simple. In Figure 5.9, we see an example in which the speaker said "and then she pirouettes" but the recognizer hypothesized "and when she peer who whets". In

[5]Note that this would be the labeling produced by the root node of any decision tree trained on the data in question. Also, to make an the analogy with language modeling, note that this labeling is the equivalent of a context-independent or "unigram" language model to predict individual word probabilities.

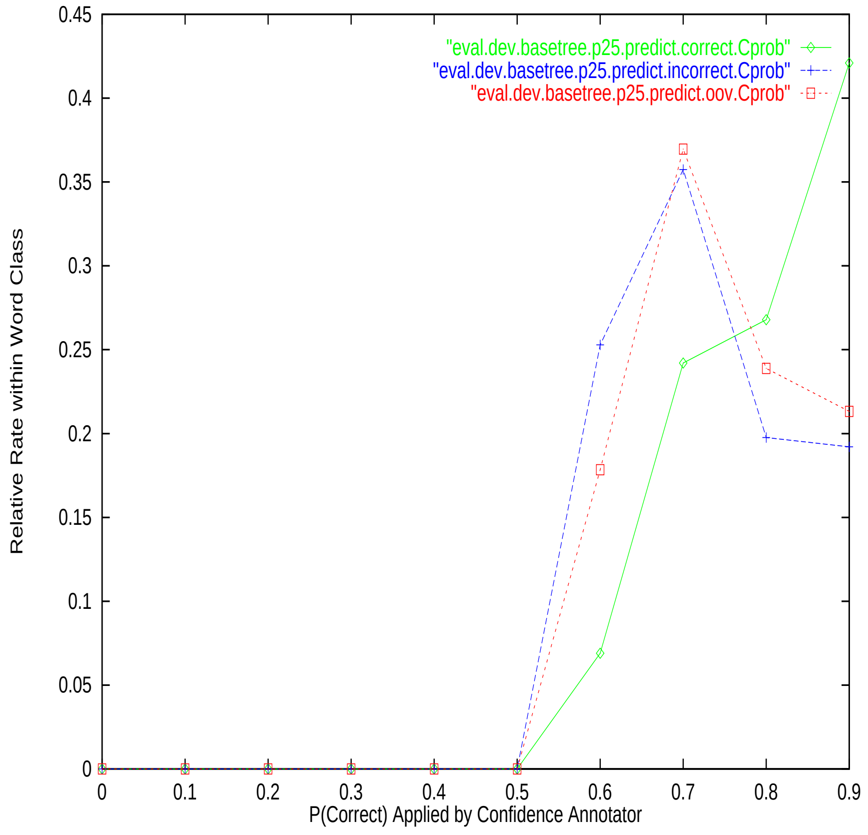


Figure 5.5: BASE1:percPhAll distribution of $P(C)$ for each of the three label classes *correct, incorrect/other, incorrect/oov* for the baseline results.

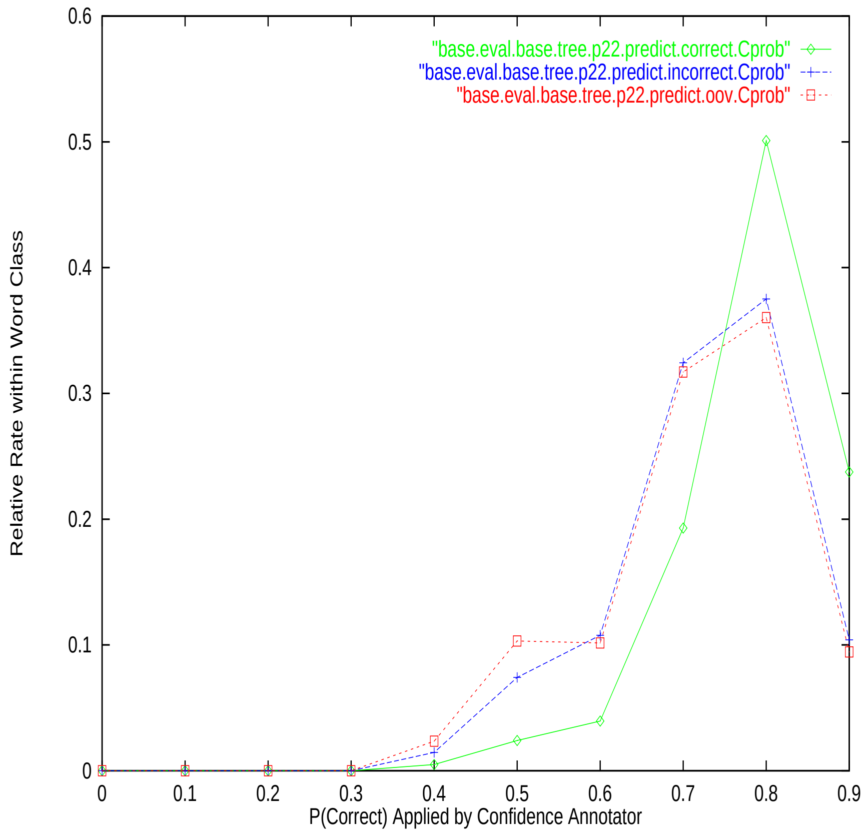


Figure 5.6: BASE2:Combined+Duration distribution of $P(C)$ for each of the three label classes *correct, incorrect/other, incorrect/oov* for the baseline results.

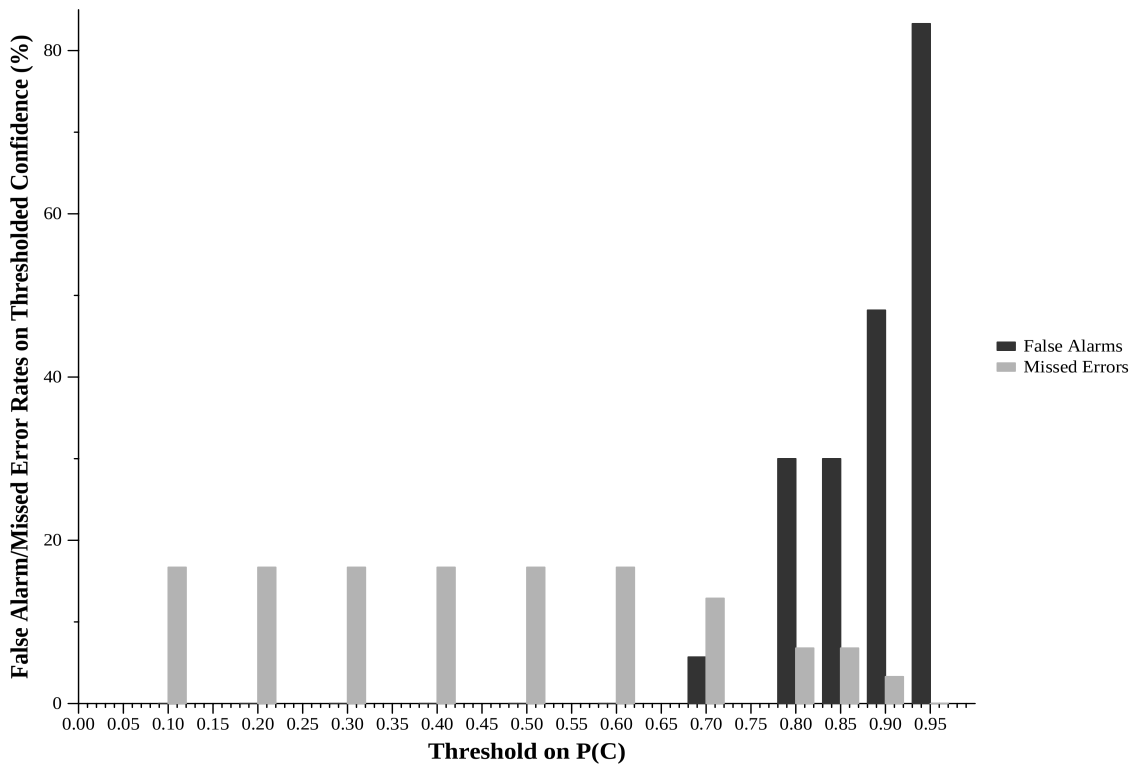


Figure 5.7: False alarm/missed error rates at various $P(C)$ thresholds for the BASE1 word-level predictor of error labels in classes $\{$*correct*,*not-correct*$\}$. NAB data.

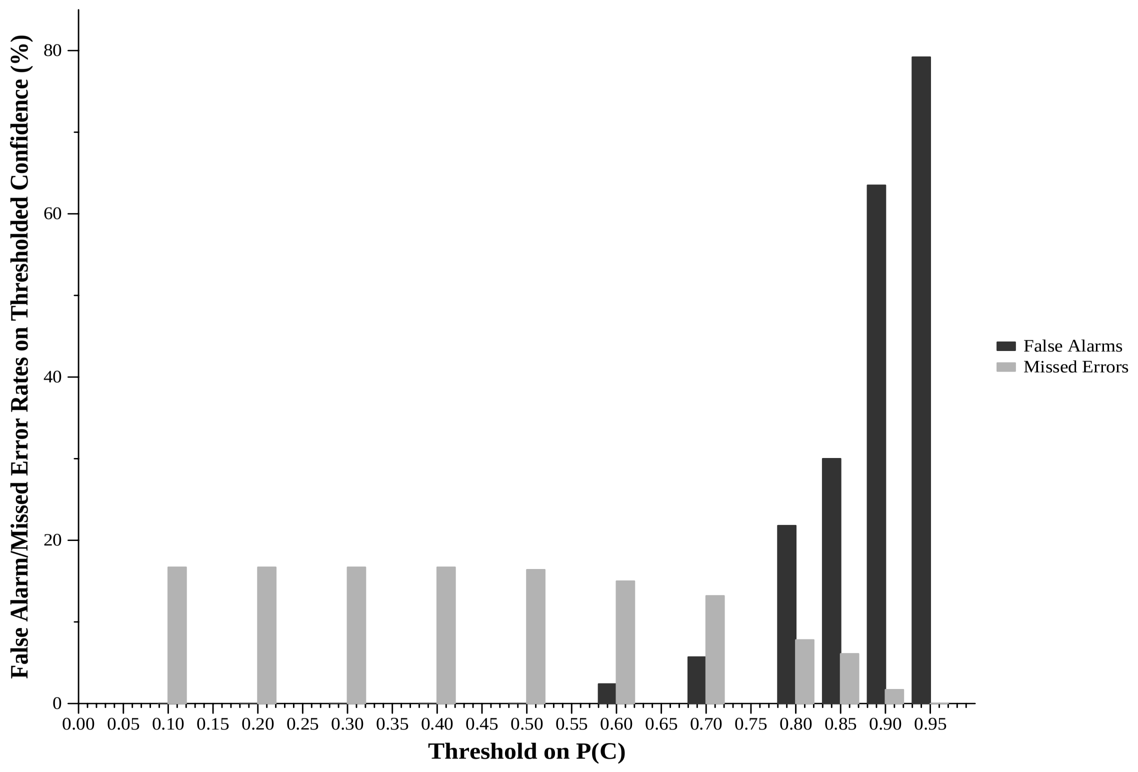


Figure 5.8: False alarm/missed error rates at various $P(C)$ thresholds for the BASE2 word-level predictor of error labels in classes {*correct*,*not-correct*}. NAB data.

this example the word “pirouettes” is out-of-vocabulary (OOV). For this reason the actual labels of the hypothesized words are:

- *Correct* for “and” and “she”,
- *Inc/OOV* for “peer”, “who” and “whets”, and
- *Inc/Other* for “when”.

To find the overall cross-entropy of the labeled stream, we make the calculation

$$CE_{sentence} = \frac{-\sum_i^N log_2(P_{actual}(w_i))}{N}$$

in which N is the length of the stream and P_{actual} is the annotated probability for word w_i that corresponds to its actual class membership. Thus in Figure 5.9 the contribution to this calculation for the first word, “and”, is taken from its $P(Correct)$, 0.9, because its actual label is “Correct”.[6]

When the logarithm is taken in base 2 the value $CE_{sentence}$ can be interpreted as representing the amount of information, measured in bits per word, remaining in the stream of confidence annotations after a particular confidence annotation model has been applied. Better annotations will have lower cross-entropy values, with a perfect scheme having a cross-entropy of zero.

Because, as described above, we are interested in comparing the cross-entropy of a stream of confidence annotations with the fundamental cross-entropy of the non-annotated output of the recognizer, we use the following expression to define our entropy-based performance measure:

$$Reduction\_in\_Cross - entropy = \frac{CE_{a\_priori} - CE_{sentence}}{CE_{a\_priori}}$$

in which

$$CE_{a\_priori} = \frac{\sum_i^N log_2(P_{a\_priori}(Correct))}{N}$$

[6]Throughout this document, the symbols $P(Correct)$ and $P(C)$ will be used interchangeably. Analogously, $P(OOV)$ and $P(Incorrect/OOV)$ will also be interchanged freely.

REF:	AND	THEN	SHE	PIROUETTES		
actual:	Correct	Inc/Other	Correct	Inc/OOV	Inc/OOV	Inc/OOV
HYP:	AND	WHEN	SHE	PEER	WHO	WHETS
P(Correct)	0.9	0.8	0.7	0.7	0.69	0.7
P(Inc/OOV)	0.05	0.01	0.01	0.1	0.01	0.2
P(Inc/Other)	0.05	0.19	0.29	0.2	0.3	0.1

$$CE = -(\log(0.9)+\log(0.19)+\log(0.7)+\log(0.1)+\log(0.01)+\log(0.2))/6$$

Figure 5.9: Calculating the overall cross-entropy of a stream of confidence annotations when compared with actual class labels.

The value $P_{a\_priori}(Correct)$ is simply the rate at which actual labels of class "Correct" appear in the stream of length N. The expression for reduction in cross-entropy indicates the proportional reduction achieved by some confidence annotator over the simplest possible guess at correctness of the hypothesized sequence.[7] Thus when a confidence annotator does not work well (its $CE_{sentence}$ is large), the reduction in cross-entropy achieved will be close to zero. When the confidence annotator performs well (its $CE_{sentence}$ is small), the reduction in cross-entropy will be close to 1.0.

The use of cross-entropy in our measure is related to the use of test-set perplexity in the analysis of language models. Consider the idea that by using our three-class labeling scheme we are essentially creating an output stream whose elements are drawn from a language that made up of three symbols. For each word in the recognizer output we pick one of the words *correct*, *incorrect/other*, or *incorrect/oov* as the next word in the sentence we're creating. Because we know we're not perfect at doing this, we want to hedge our bets a little, giving probabilities to each instead of putting all our money on one language element at each choice point. By using perplexity as a measure of how well we do at creating this sentence of error class labels (compared to the correct labels) we can get a sense for how good we are at placing these bets. If we do a perfect job, laying all of our money down on one choice only at each word, and getting them all correct, then our perplexity will be of value "1". If we lay even odds at each word, guessing $1/N$ as the probability for each of N classes, then our perplexity will be N. The perplexity tells us, on average, how many choices were considered from the language as the correct selection at each point. The actual expression for perplexity is related to our expression for cross-entropy by:

$$PP_{sentence} = 2^{CE_{sentence}}$$

In general we make the calculations of cross-entropy over the entire test corpus[8] as one sentence, ignoring utterance boundaries.

One thing to note is that the reduction in cross-entropy metric can "blow up" when the confidence annotator being measured produces guesses that

[7]Throughout this document we typically multiply this number by 100fractions.

[8]The confidence annotator described here performs remarkably better on silence and noise words than it does on lexical items. In order to give a fair measure of the baseline performance I chose to ignore this class of items and focus on lexical items. All numbers reported in this chapter and all chapters relating to word-level annotation exclude silence and noise output tokens.

give a zero probability to one or more classes. This would not be a problem if all such predictions were accurate, but in the case of wrong guesses of this type we are faced with asymptotic behaviors in the performance metric. Thus it is usually advisable to do some kind of smoothing within predictors measured under the reduction in cross-entropy metric.

For the baseline system we smooth the guesses of the confidence annotator with the uniform probability distribution, giving the uniform distribution a weight of $0 < \lambda \leq 1$. We can pick an optimal value of λ for each combination of confidence annotator and recognition task. In this case a λ of 0.01 works well.

The overall performance of our two baseline systems are presented in the first column of Table 5.1. They are very similar.

5.4.2 The Cross-entropy of Multi-class Labelings (ν)

What if we want to look a little more closely at these results, analyzing the confidence annotator's performance on each class label instead of its overall behavior? One natural extension of the reduction in cross-entropy measure is to use it to measure the entropy of an annotation with respect to a particular class. To accomplish this we simply collapse the correct labelings of the test data into two classes from the original N. In the case of the baseline, this means that if we want to examine the annotator's performance in labeling class *correct*, we modify the "true" labels from the original *correct*, $incorrect/other$ and $incorrect/oov$ to *correct* and $not - correct$. All of the truth labels $incorrect/other$ and $incorrect/oov$ are changed to $not - correct$ in the transcript. All of the probability scores in the annotation for classes $incorrect/other$ and $incorrect/oov$ are summed together and become the scores for the new class $not - correct$. After these two transformations have been made the same perplexity and cross-entropy calculations are performed. This same collapse into "class X" and "class not X" can be performed for any of the N classes. A simple example of how the calculations look for the baseline case of $N = 3$ appears in Figure 5.10.

As we can see in Table 5.1, the two baselines no longer look so similar after breaking out their performance by class. In particular, BASE2:Combined+Duration does much better at finding errors due to OOVs than does BASE1:percPhAll. At the very least, this means that when the recognizer is trying to cover OOV words it is choosing words that are made up similarly to the phone-only decoding. Thus the percPhAll predictor is not helping to find this class of

REF: AND THEN SHE PIROUETTES

actual:	Correct	Inc/Other	Correct	Inc/OOV	Inc/OOV	Inc/OOV
HYP:	AND	WHEN	SHE	PEER	WHO	WHETS
P(Correct)	0.9	0.8	0.7	0.7	0.69	0.7
P(Inc/OOV)	0.05	0.01	0.01	0.1	0.01	0.2
P(Inc/Other)	0.05	0.19	0.29	0.2	0.3	0.1

CE(Correct) = -(log(0.9)+log(0.2)+log(0.7)+log(0.3)+log(0.31)+log(0.3))/6
CE(Inc/OOV) = -(log(0.95)+log(0.99)+log(0.99)+log(0.1)+log(0.01)+log(0.2))/6
CE(Inc/Other) = -(log(0.95)+log(0.19)+log(0.71)+log(0.8)+log(0.7)+log(0.9))/6

Figure 5.10: Calculating the cross-entropy of confidence annotations. Each class is considered separately, using the collapsing scheme described above.

Confidence Annotator	% overall	% cor	% incor	% oov
BASE1:percPhAll	5.1%	6.3%	5.9%	2.5%
BASE2:Combined+Duration	5.3%	6.0%	4.5%	5.2%

Table 5.1: Performance of the two baseline systems overall and broken out by category. Measured in percent reduction in cross-entropy.

error.

5.4.3 Characteristics of the Cross-Entropy Measure (ν)

The cross-entropy measure is a well-founded means of evaluating confidence annotations. It is also flexible, as illustrated by these points:

- It is straightforward to use in evaluating simple "right/wrong" types of labeling.
- It extends naturally for use in evaluating multi-class labeling.
- It allows different confidence annotators to be compared in the same task domain.[9]
- It allows a confidence annotator that is capable of labeling all types of errors to be compared directly with a confidence annotator that produces annotations only for a subset of the classes.[10]
- Within limits this method can also serve as a fair comparison across tasks. If two different task domains have somewhat similar baseline class experience rates then no unreasonable behaviors in the metric arise.[11]

[9]As long as relative improvement over self-entropy of the recognizer output is considered instead of absolute performance in bits per word.

[10]As long as the experience rates of the classes in the recognizer outputs of the systems being compared are similar for the subset of classes in question.

[11]If, however, one of the task domains has either extremely good or extremely poor word error performance then it is possible to construct confidence annotators whose guesses are not easy to interpret under this metric. Consider, for instance, a domain in which a recognizer has 99% accuracy. In this case there's not much room for improvement above the "0.99 correct" guesses the "unigram" model would make. Thus it wouldn't be fair to

These characteristics combine to recommend the cross-entropy measure as a generally useful metric in evaluating confidence annotation. However, while the cross-entropy measure is good at indicating how well a confidence annotator performs overall, it has two weaknesses. First, it is not an intuitively appealing metric in all situations. To correct this problem it is best used in conjunction with probability curves of the type shown in Figures 5.5 and 5.6 or some other graphing method which will help give intuitive insight. These figures are graphs of the probability of word membership in class *correct*, $P(C)$, which is assigned as one of the three probabilities given to each HYPothesized word by the confidence annotator. The x-axis represents the $P(C)$ given to a word. The y-axis indicates the relative rate at which this probability was applied within each of the three word classes. Looking at these curves and comparing the two systems gives us insight into exactly why the BASE2:Combined+Duration cross-entropy reduction figure was so much lower for class *incorrect/oov* than for BASE1:percPhAll. In general we will report these curves alongside our main cross-entropy reduction figures. In addition to these $P(< class >)$ curves we will also report the tradeoffs between false alarms and missed errors that occur when various thresholds are applied to the curves. For example, in Figure 5.7 we see the effect of sliding a threshold $P(C)$ from zero to one along the curve for class *correct* in Figure 5.5. Any of the annotated words tagged with a $P(C)$ value above the threshold will be labeled as "correct". Any below will be labeled as "incorrect". If we compare those labeled "correct" and "incorrect" with those actually in classes *correct* and $not - correct$, we get a 2-by-2 matrix that defines a set of correct labels, false alarms, and missed errors. (See Section 6.9.1 for a detailed description of the calculation of this matrix.) As the threshold is increased the missed errors decrease, at the cost of a rising false alarm rate. We can compare the tradeoffs given between false alarms and missed errors by any two confidence annotators as an additional measure of their relative effectiveness. In general we report these tradeoffs wherever $P(< class >)$ curves are reported.

On another note, the cross-entropy reduction metric is not always good at evaluating the quality of very specific or focused instances of labelings. For instance, even though our baseline predictor shows good improvement in ability to predict the class *incorrect/oov*, it may well be the case that

compare an annotator's efforts in this domain with one in which the base accuracy rate is 70%.

individual cases of OOV words in the speech being analyzed could be completely missed. In order to truly evaluate an annotator's ability to pick out individual OOV words, a task-related metric such as ROC curves on detections would be more appropriate. (See the end of Chapter 6 for an example of this.) Thus in each of the experiments reported later in this document in which confidence annotators are used in a particular domain I will report both cross-entropy and application-specific performance figures.

5.5 Summary

After motivating the potential usefulness of confidence annotation, we discussed the necessary design elements of confidence annotators, as well as a set of methods for evaluating their performance. Two baseline systems were presented along with performance descriptions and data.

Chapter 6

Word Confidence Annotation for Large Vocabulary Read Speech

Typically speech recognizers output a single best guess at the sequence of words uttered by a speaker. Applications that rely on speech recognizer output usually integrate this output directly, without benefit of additional information such as confidence annotations. For example, the word sequences output by a speech recognizer are commonly used directly to tag the voice components of multi-media databases for retrieval [21]. In another example, B. Suhm [58] [57] has found in his work on dictation error correction that the unannotated use of recognizer output places the burden of detecting errors in the recognizer output squarely on the shoulders of the user of the dictation system.

The usefulness of automatic speech recognition in applications such as multi-media databases and dictation systems can be improved by the extension of speech recognizer functionality to include confidence annotation at the word level. This additional layer of "meta-information" can be used to filter the output of the recognizer into regions of high and low reliability. In the case of multi-media databases this can improve the quality of tags used for retrieval. In the case of an editing interface to a dictation system the user's attention can be focused on suspect regions in the output. As described later in this chapter, the interpretation of confidence annotations can be adjusted in a way that allows a user to trade off the rate at which errors in slip past unnoticed against the rate at which output words are falsely labeled as being incorrect.

In this chapter we describe a set of useful predictors of error for the Sphinx-II system applied to a speech-to-text-dictation task. We give detailed descriptions of how each of these predictors behave in identifying words as being in one of three classes:

1. *correct*: the speech recognizer guessed the correct word with a frame segmentation within two frames of its location in the "ground truth" definition, or

2. *incorrect/oov*: the speech recognizer could not guess the correct word because the word was not in the recognition lexicon, thus one or more errors of substitution were made, or

3. *incorrect/other*: the recognizer had the possibility of finding the correct word but did not, instead either guessed the wrong word or the right word at a frame segmentation more than two frames from its "true" location. [1]

The predictor variables are constructed from several important sources of information. Some of these come directly from the internal workings of the usual speech recognizer itself. Others are additional "instrumentations" computed in parallel. These various sources of information are described in detail in Chapter 2. They include:

- Word and phone segmentations and acoustic scores from the best-scoring HYPothesis produced by the recognizer,

- Language model score and calculation source information from the same best-scoring HYPothesis,

- Characteristics about the words HYPothesized, including their pronunciations and how frequently they appear in acoustic training materials,

- The results of a parallel "phone-only" decoding, in which the recognizer is not constrained to either phone sequences from the dictionary or word sequences in the language model in its recognition of phone sequences,

[1]In order to be considered as properly segmented, both the start and end frames of a word must be within two frames of the *correct* location. Thus a word in class *correct* might actually be four frames shorter or longer than its *correct* decoding without being considered as $not - correct$.

- The results of a completely unconstrained frame-by-frame decoding in which the best possible acoustic score and basephone are determined for each frame of the utterance,
- Three distance metrics between basephones, including a simple match count at the frame level, a phonologically-based similarity measure (H_{WC}), and an empirically derived confusion-based distance measure, and
- The contents of an N-best list that contains complete word and phone segmentation and score information for each of 150 elements.

After constructing and analyzing each predictor variable from these sources we compare various combinations in their ability to work together to assign probabilities of error class membership to words hypothesized by the recognizer. A variety of subgroupings are compared using the reduction in cross-entropy measure developed in Chapter 5. The collection of predictor variables that performs best under this metric are then evaluated, with the performance results reported in several ways:

- overall reduction in cross-entropy of labelings,
- class-by-class reduction in cross-entropy,
- visual comparison of probability distributions for membership in class *correct* across the three actual classes, and
- a comparison of false alarm/missed error rates at various threshold points on the probability-of-correctness curve.

All of these measures report essentially the same information. They are supplied to assist each reader in finding the most immediately accessible form for his or her own purposes.

Finally, domain-specific performance measures are discussed in order to illustrate the importance of using domain-dependent measures when evaluating confidence annotators.

The data used in these experiments is the NAB data described in Chapter 3. The curves shown for each predictor variable come from the development data, which serve as training data for the confidence annotator. All four above forms of reporting confidence annotation results are given for the independent evaluation test data.

We will now turn to descriptions of individual predictor variables. Each predictor's computation is described, and graphical representations of its ability to separate error classes are provided. Quantitative results that compare predictor performance levels appear in the section after this.

6.1 Number of Phones (ν)

The simplest predictor variable (after those included in the baseline systems of Chapter 5) is the number of phones found in the word's dictionary pronunciation. [2]

In Figure 6.1 we see this variable's distribution across the word classes *correct*, *incorrect/other*, *incorrect/oov*. The x-axis of this figure is the number of phones found in the word. The y-axis indicates the relative rate of the variable in a particular class.

This predictor helps separate class *incorrect/oov* from the other two classes – on average the recognizer prefers to use slightly longer words than average to "cover" OOV words in the utterance.

6.2 Word Rate in Acoustic Training

Another simple predictor variable is the count of the number of times a word was seen in the acoustic training material.[3] Figure 6.2 shows the distribution of this variable across the three word classes. The predictor manages a slight distinction between *incorrect/oov* and $not-incorrect/oov$ words. The recognizer has a slight preference for using words that were not seen in the acoustic training data somewhat more frequently for decoding *incorrect/oov* words than for other classes.

[2]By default, throughout this chapter "word" refers to a single pronunciation variant of a word.

[3]The use of this variable was prompted by its use at BBN [12] in a different but related context. When diagnosing errors using known transcripts of spontaneous telephone conversations, it was found that in general words seen in acoustic training were more reliably decoded than unseen words. Here we're not using this predictor for diagnosis, but prediction. This distinction actually changes the interpretation of the results pretty radically. Whereas in the BBN case the interpretation was simple – if you haven't trained on a word you probably won't decode it right – the interpretation is not so clear in this case.

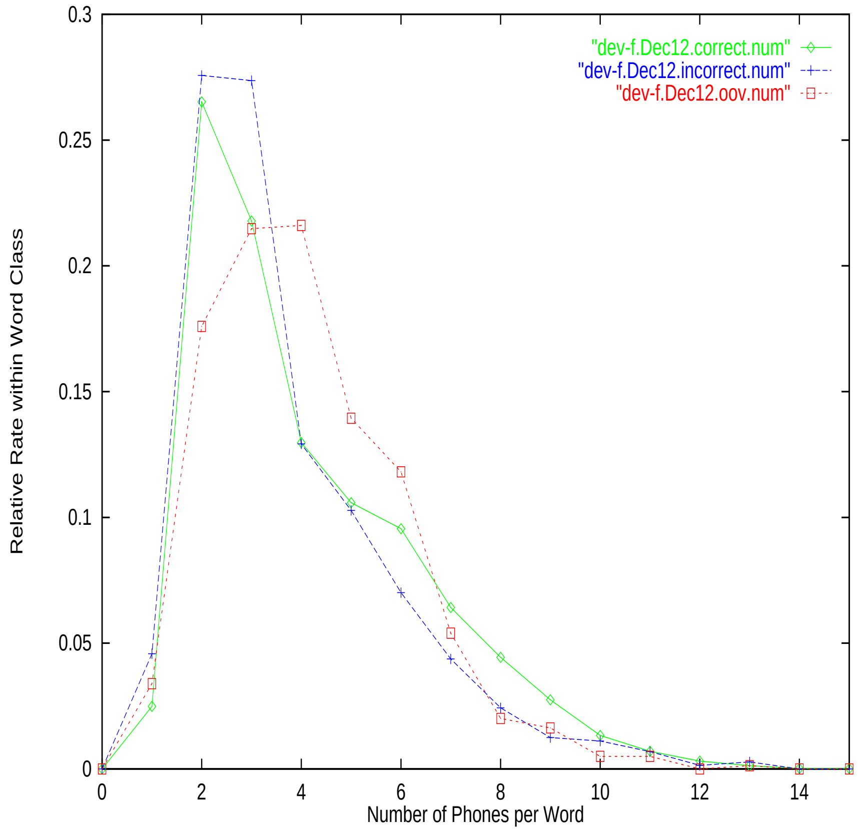


Figure 6.1: Values of number of phones per word broken out by word classes $\{correct, incorrect/other, incorrect/oov\}$.

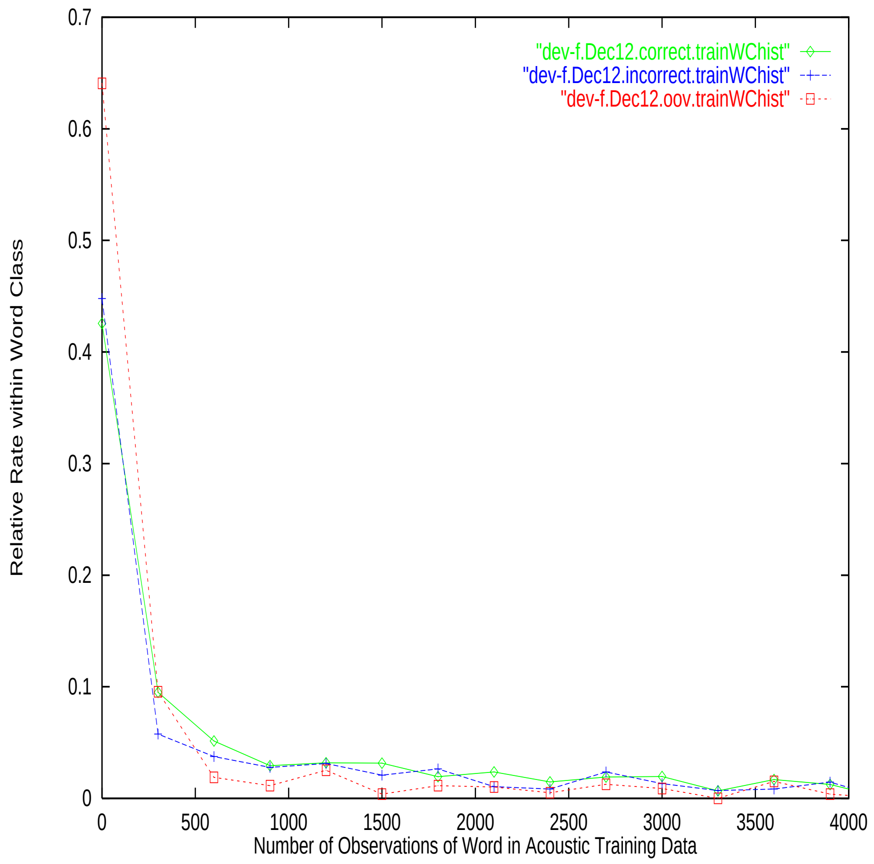


Figure 6.2: Number of times word was seen in acoustic training data broken out by word classes $\{correct, incorrect/other, incorrect/oov\}$.

6.3 N-best List Characteristics

There are two related predictor variables we use for looking at the "spread" of possible choices present at any given point in time in the N-best list. We will examine each in turn.

6.3.1 N-best List Homogeneity

The best predictor of whether a word is correct or not is a measure I call "N-best list homogeneity". The purpose of this measure is to identify regions in the utterance in which the recognizer is not completely sure what the best answer might be. It is calculated for each word in the HYPothesized sequence by searching the N-best list of the utterance for all instances of the same word that are segmented within a tolerance of two frames from the best guess. The ratio of matches found to the total number of entries in the N-best list is calculated, with each found instance weighted by the total path score of the N-best list entry in which is was found.[4]

The implementation of this predictor variable follows that reported by BBN in [29]. This predictor variable is directly related to strong predictors reported by [51], [41], [20], and [5]. It captures information about how many distinct possibilities[5] the recognizer was considering over the frames included in the hypothesized word's segmentation.

In Figure 6.3 we see the distribution of values of N-best homogeneity for words that were actually in classes *correct*, *incorrect/other*, *incorrect/oov*. The x-axis in this graph is the value of the N-best homogeneity score, which varies between zero and one. The y-axis is the relative rate of occurrence of the N-best homogeneity score for each word class. The curve that begins near $(0, 0)$ and ends near $(0.95, 0.12)$ is the curve for N-best homogeneity scores assigned to words actually in class *correct*. The point at $(0.8, 0.04)$ is interpreted to mean that 4% of words actually in class correct were assigned N-best homogeneity scores between 0.8 and 0.85. The other two curves on the graph give the distribution of values for words actually in classes *incorrect/oov* and *incorrect/other*.

We see that:

[4]Both weighted and unweighted versions of this algorithm were tried. The resulting differences between these were very small.

[5]An exact count of words and phone, and an estimate of individual states...

1. N-best homogeneity scores are higher for *correct* words than *not − correct* words,

2. the N-best homogeneity predictor separates *correct* words from *not − correct* words, but doesn't help split words wrong due to OOVs from those wrong for other reasons, and

3. the primary separating breakpoint for this predictor is just below the value 0.8.

6.3.2 N-best Average Word Rate (ν)

A related predictor variable is based on the distinct word count in the N-best list at any given frame. The purpose of this predictor is the same as for the previous one: we seek for regions in which the recognizer is confused about what the right answer might be. First the number of unique words present in the N-best list is calculated for each frame of the utterance. This value is then averaged over the frames of the hypothesized word segmentation.

The resulting curves are shown in Figure 6.4. The x-axis in this graph is the average number of words present in the N-best list at the same point in time as the word in question. The y-axis indicates the relative frequency of words with the corresponding x-value for a specific class of word. The point at $(1, 0.62)$ should be interpreted to mean that 62% of words actually in class correct had between 1 and 2 words, on average, present in the N-best list in addition to themselves.

This predictor variable shows a clear distinction between *correct* and *not − correct* classes, but as we will see later in this chapter, it is not nearly as strong a predictor of error as the conceptually similar N-best list homogeneity score of Figure 6.3.

6.4 Language Model Predictors (ν)

We can measure two aspects of the language model's contribution to the selection of a given HYPothesized word. The most useful language-based predictor is the raw language model score. Of secondary but non-negligible importance is the "language model source" predictor (described in Section 2.4), which captures the details of how the language model score was produced.

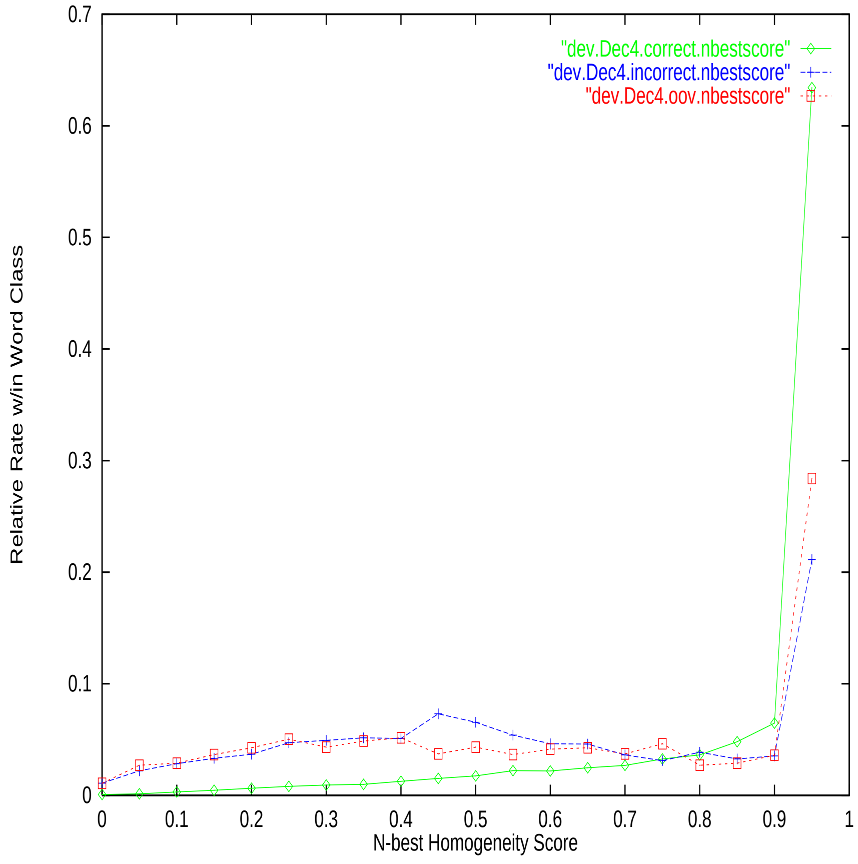


Figure 6.3: N-best homogeneity scores for words of NAB devtest in {*correct, incorrect/other, incorrect/oov*}.

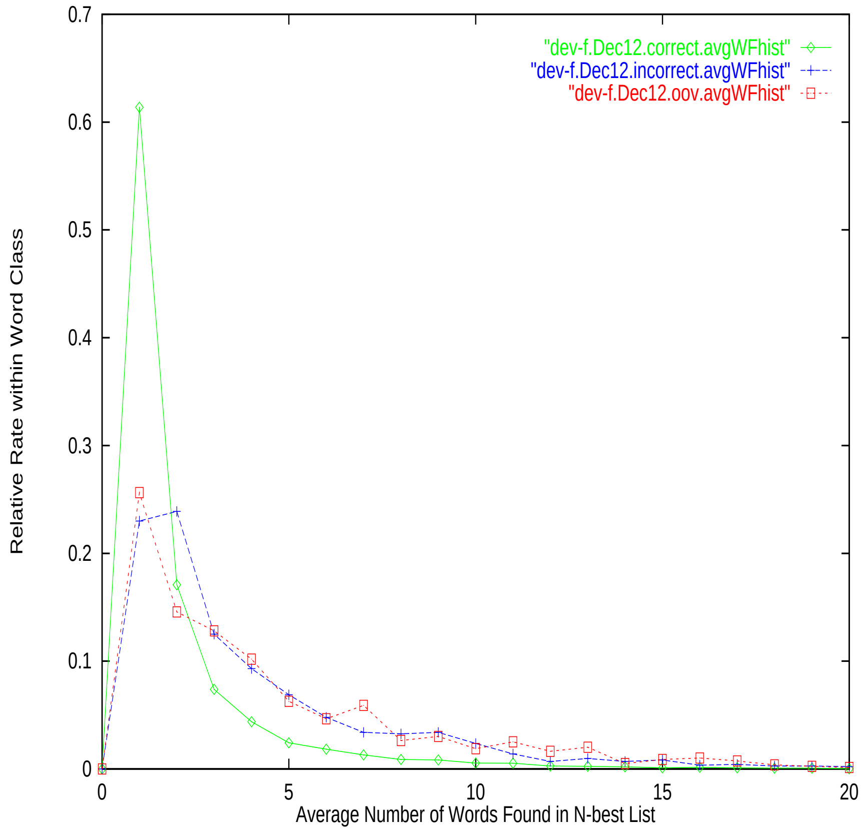


Figure 6.4: N-best average word frequency scores for words of NAB devtest in $\{correct, incorrect/other, incorrect/oov\}$.

6.4.1 Language Model Score

In Figure 6.5 we see the distribution of language model scores across our three word classes. The x-axis of this graph represents the negated raw language model score, with the more likely scores at the left and less likely values at the right. The curve that has a sharp peak at $(1.2 \times 10^6, 0.1)$ is the curve for words that are actually in class *incorrect/oov*. The peak point in this curve is interpreted to mean that 10% of all words in class *incorrect/oov* have a raw language model score between -1.2×10^6 and -1.25×10^6. The other two curves, which are very similar to each other, are for words actually in classes *correct* and *incorrect/other*.

From this graph we see that:

1. language model score values help separate errors due to OOV words from other kinds of errors, and

2. *correct* words get better language model scores than $not - correct$ words.

6.4.2 Language Model Source (ν)

As described in Section 2.4, there are five branches in the algorithm that is used to create our trigram language model scores. Table 6.1 summarizes how these five cases are distributed in the NAB development data over the actual word classes *correct*, *incorrect/other*, *incorrect/oov*.

Note that the relative rates of the use of trigrams drops from its peak in class *correct* to its lowest in class *incorrect/oov*. complimentarily, the rate at which backing off all the way to unigrams (BU) is used is lowest for correct words and highest in class *incorrect/oov*.

6.5 Word Acoustic Predictors (ν)

Unfortunately, the simplest word acoustic predictor, duration-normalized word acoustic score, is not a good predictor of error. To get information about errors from acoustic-related problems at the word level we need to look at more detailed measures of acoustic performance. In this section we will examine several schemes that can be loosely considered as "acoustic normalization" techniques. The first is actually a score normalizing approach.

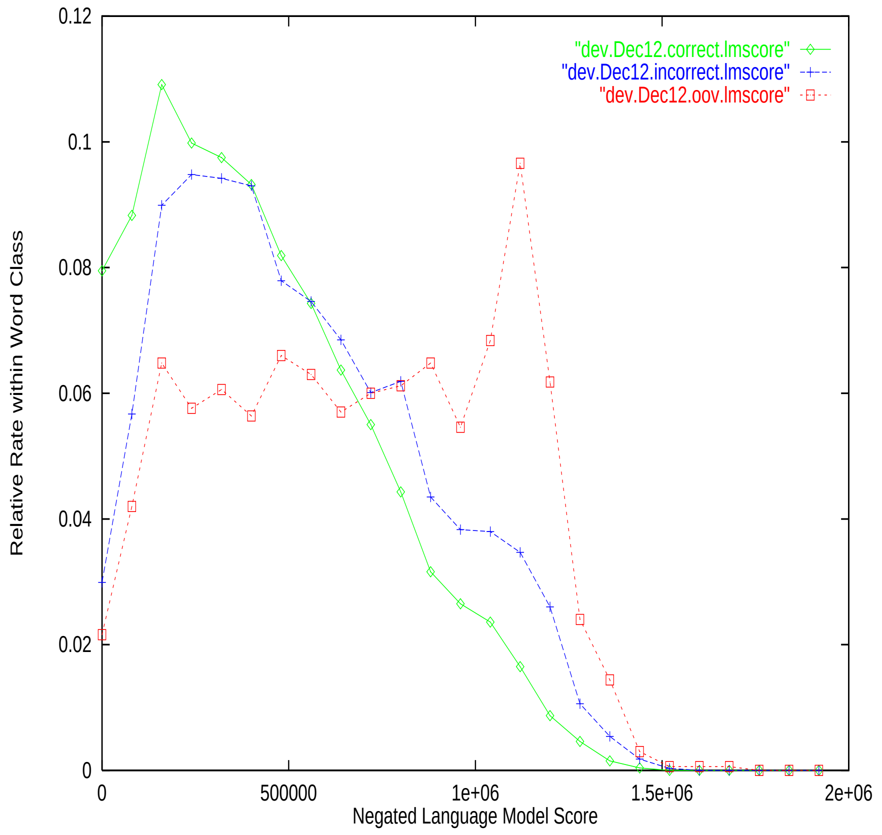


Figure 6.5: Negated language model scores for words of NAB devtest in $\{$*correct, incorrect/other, incorrect/oov*$\}$. Values close to zero correspond to more likely word predictions. Values close to 2,000,000 are extremely unlikely word predictions.

LM Backoff Branch	%overall	%correct	%incorrect	%oov
T	58%	61%	46%	32%
BB-B	22%	20%	30%	31%
B	15%	15%	16%	16%
BB-BU	4%	3%	6%	15%
BU	1%	1%	2%	6%

Table 6.1: Rates of $\{correct, incorrect/other, incorrect/oov\}$ within the five branches of the Katz LM calculation for NAB development tests.

The other approaches that follow involve moving the "normalization" process into various basephone descriptor spaces. We seek leverage in the comparison of the hypothesized phone sequences against various "best guess" basephone sequences.

As described in Chapter 2, in Sphinx-II there is a unique mapping from senone to basephone. Thus, loosely speaking, we "know" the "best basephone" at each frame in an utterance. We can also guess at the best basephone at a given frame by looking at the phone-only decoding. One way to analyze error behavior is to compare the phone sequence in the hypothesized words with our two guessed best basephone sequences. These comparisons are accomplished through a variety of distance metrics that make scalar measurements of the form

$$< basephone > \times < basephone > \rightarrow < distance >$$

We have already seen an example of such a distance measure in the baseline system reported in Chapter 5 – the percentage of phone matches at the frame level between the HYPothesized phones for a word and the phones found in the phone-only decoding were used in the simplest baseline. In a loose sense, the phone-averaged score normalization described in the previous section measured a distance between the phones in the HYPothesized word and the basephone that corresponds to the best senone score at each frame. In the following sections we examine several distance measures between basephones and their relative performance in predicting word class membership when applied to our two "best basephone guessing" methods.

6.5.1 Best Senone Score Normalization (ν)

The first acoustic normalization scheme is suggested by a portion of the fundamental equation of speech recognition [30]:

$$P(w|A) = \frac{(P(w) \times P(A|w))}{P(A)}$$

In practice we typically ignore the term $P(A)$ in the denominator, since for all of the difference possible decodings for a particular utterance this value is constant.

But for the purposes of determining the relationship between the value

$$\frac{P(A|w)}{P(A)}$$

for a word w and the correctness of w, the use of some $P(A)$ as a normalizing factor for our word acoustic score $P(A|w)$ can work well. (This approach is similar to that found in [10].)

In related experiments involving (among other things) the automatic identification of errors in conversational speech recognition, Young and Ward [65] used a similar approach. Their estimate of $P(A)$, which we will call $P(A)_{YW}$, came from scores derived from a phone-only decoding done in parallel with the word-level recognizer. (The phone decoding used a bigram phone model to guide search.) When performing word decoding, $P(A|w)$ is calculated under phone sequence constraints from a dictionary. By comparison, the value $P(A)_{YW}$ was calculated by relaxing these constraints one level: one phone could follow any other phone, but within-phone constraints were still enforced. Thus $P(A)_{YW}$ could be said roughly to represent the probability of the acoustics given American English phonological constraints.

In this thesis, our estimate of $P(A)$ is calculated in a one-step-more-relaxed fashion. We do not enforce phone-internal state sequence constraints at all. Instead we allow any state to follow any other, and construct a $P(A)_{SS}$ from the combination of the best senone score at each frame within the segmentation of w. These values are calculated in the manner described in Section 2.8.2. In comparison with the Young and Ward estimate, our less-constrained version of the estimate of $P(A)$ can be said to represent the probability of the acoustics given sounds generated by an American English speaker.

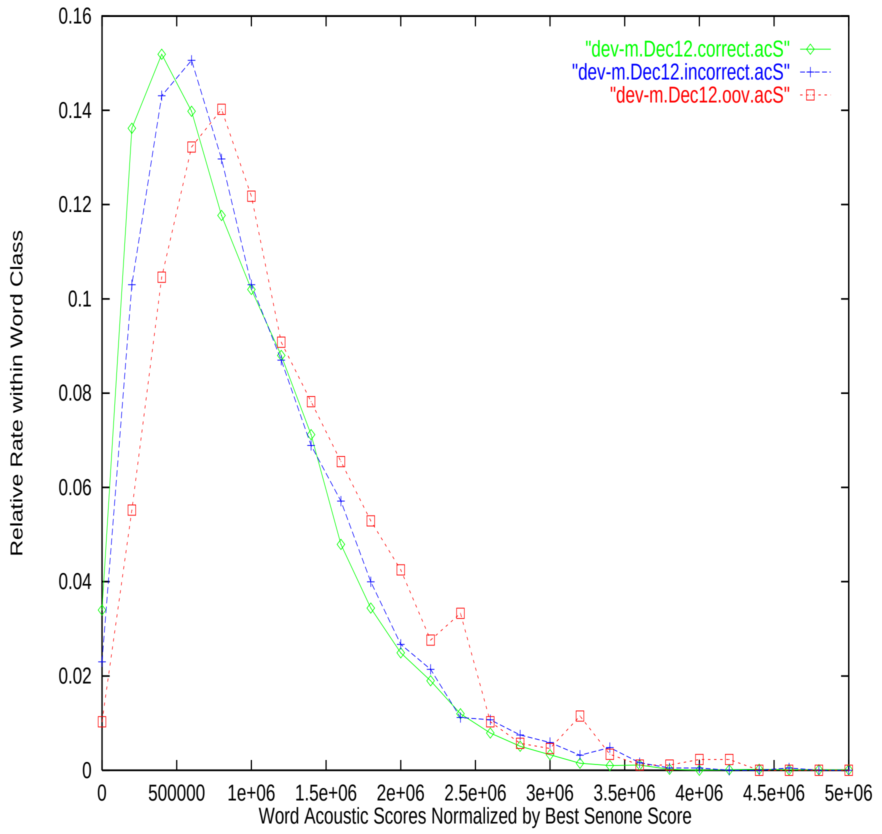


Figure 6.6: Best-senone-score-normalized acoustic word score curves for each class label in {*correct,incorrect/other,incorrect/oov*} from the NAB development data. Male models.

For words whose best dictionary-constrained decoding is very similar to the unconstrained state-level decoding, the ratio $P(A|w)/P(A)$ will be close to one(1). For words whose constrained match is not as good as the unconstrained match, this ratio will decrease. In practice acoustic scores are calculated and used in the log probability domain. Thus as the ratio decreases so does the difference between the $logP(A|w)$ and $logP(A)$. Figure 6.6 shows the difference values for the words in the NAB development data using the $P(A)_{SS}$ normalizing factor, broken out by the class labels $correct, incorrect/other, incorrect/oov$. The values in this graph are calculated frame-by-frame over entire words, ignoring word-internal phone boundaries.

The x-axis of this figure represents the difference between the raw and normalized word acoustic scores. The y-axis indicates the relative frequency with which words belonging to a given class are seen with the value on the x-axis. The leftmost curve with a peak at $(0.5 \times 10^6, 0.155)$ shows the distribution of values for words in class *correct*. The middle curve is for words in class *incorrect/other*. The rightmost curve is for class *incorrect/oov*. There is a slight but not perfectly clear separation between the *correct* and $not - correct$ distributions.

If, however, we do this normalization on a phone-by-phone basis and then average the resulting values within the word, as in Figure 6.7, we see a clearer separation between the *correct* (leftmost and tallest) and $not-correct$ (remaining two) curves.

6.5.2 Best-Senone Basephone Distances (ν)

In this section we will examine two distance metrics applied to the basephone guessing method based on the best senone scores at each frame. Section 2.8.3 describes a distance metric based on Withgott and Chen's phonological descriptors. When this distance is calculated for each phone in the HYPothesized word as compared to the best-senone basephone at each frame, the results are as in Figure 6.8. Here the H_{WC} distance value has been calculated for each phone separately, then averaged within the word. The x-axis of this figure is the H_{WC} value. The y-axis is the relative frequency of the values for a given word class. The leftmost and tallest curve is for class *correct*. There is a slight but not completely clear separation between the *correct* and $not - correct$ classes.

We can compare these results with those of a similar calculation – in the

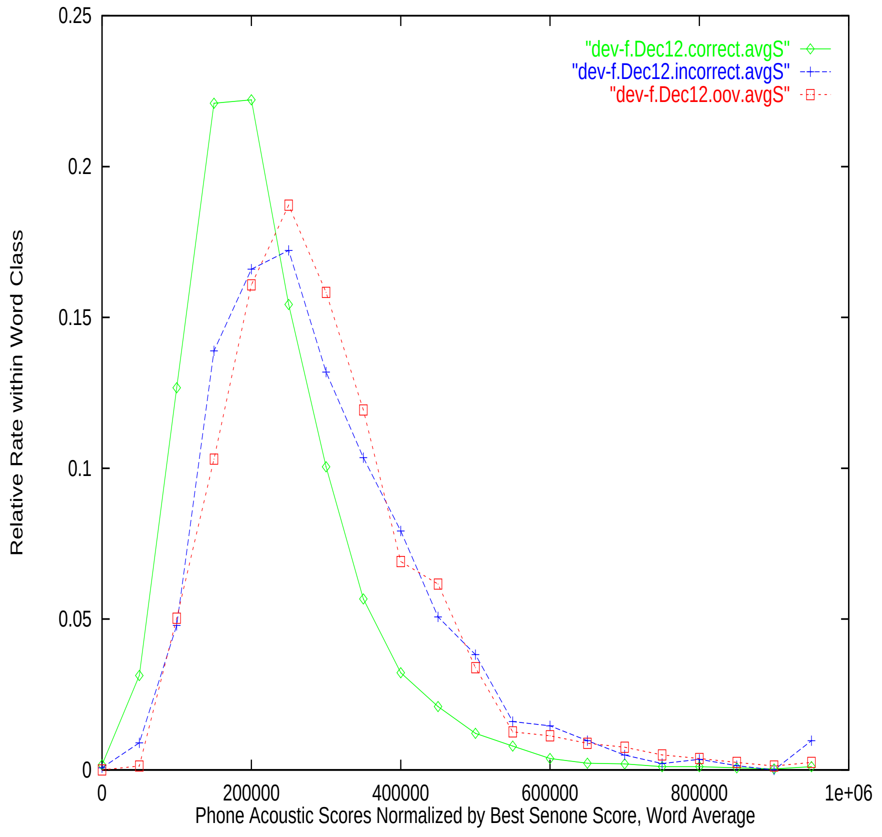


Figure 6.7: Best-senone-score-normalized acoustic word score, calculated phone-by-phone and then averaged within word. Curves are for each class label in $\{correct, incorrect/other, incorrect/oov\}$ from the NAB development data. Female models.

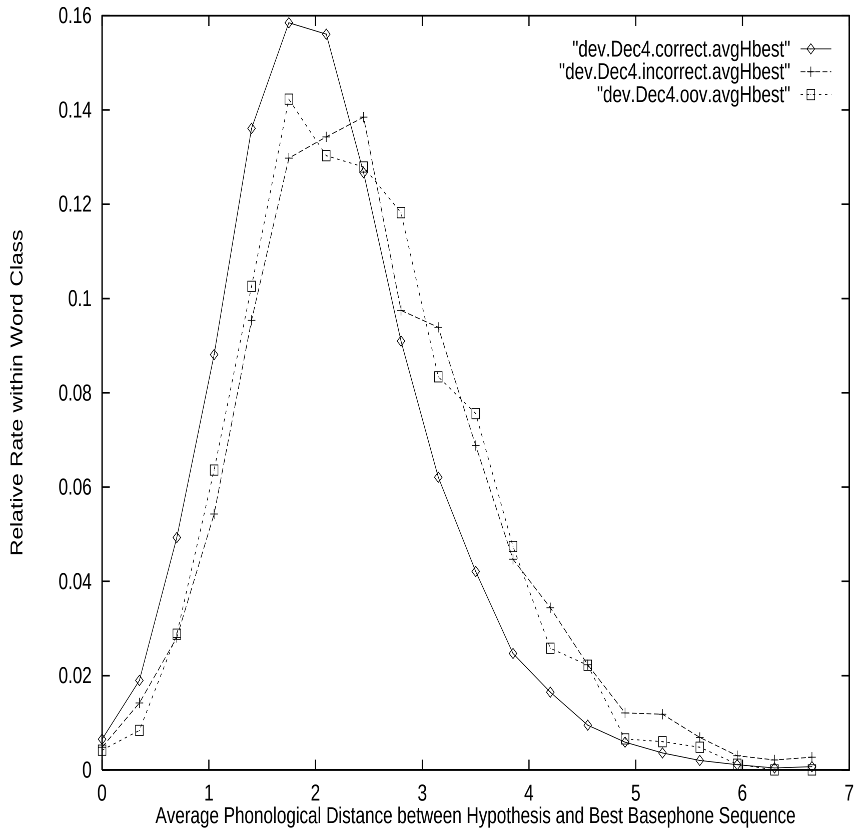


Figure 6.8: Average Hamming/WC (H_{WC}) distance score between hypothesized word and best-senone basephones. Curves shown are for each class label in {*correct, incorrect/other, incorrect/oov*} from the NAB development data.

new case the H_{WC} values are calculated for each phone and the maximum value for a phone within the word is reported. Unfortunately, using the *max* instead of the *average* removes what little information was present in the first case, as the curves for each class are essentially indistinguishable. Similarly uninteresting results are obtained when the confusion-based distance metric described in Section 2.8.3 is applied to the best-senone basephone guess. (No graph is presented.)

6.5.3 Phone-only Distances (ν)

In this section we apply the same two distance metrics, but use the phone-only guessing method as our reference point for comparison with the HY-Pothesized phones. Figure 6.9 shows the results of applying the H_{WC} metric averaged over the word. (This is analagous to the distributions found in Figure 6.8, but with an alternate best basephone guessing mechanism.) Comparison of the two figures shows that using the phone-only decoding as the reference provides a much-improved separation between words in class *correct* and other words.

When the *max* is used instead of the *average*, however, we again see the elimination of useful separation. Application of the confusion-based distance metric also does not yield a clearly separate set of curves. (No graphs are presented.)

Based on visual inspection of these graphs it appears that for this combination of speech recognizer and task that:

1. the phone-only best basephone guessing method is more reliable than the method based on the best-senone score at each frame,

2. the H_{WC} distance metric works better than the confusion-based metric,

3. only one combination actually gives a good separation between the $correct$ and $not - correct$ words, and

4. none of the combinations give good separation between words in class $incorrect/oov$ and other words.

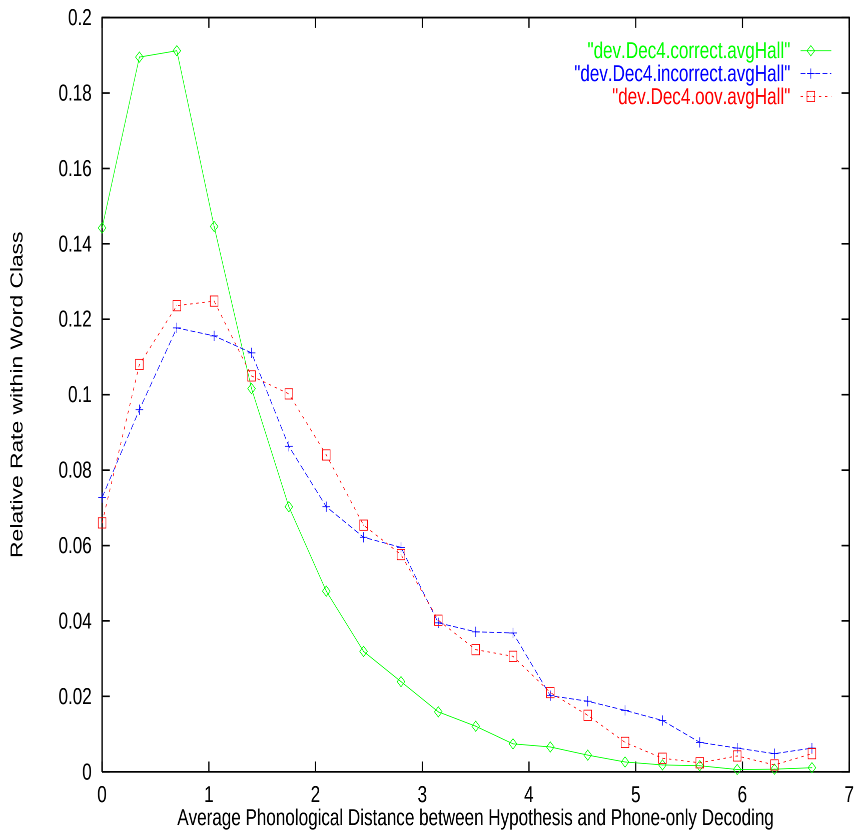


Figure 6.9: Average Hamming/WC (H_{WC}) distance between phone-only basephone and HYPothesized word phones. Curves shown are for each class label in *correct, incorrect/oov, incorrect/other*. NAB development data.

6.6 Senone Rank (ν)

As described in Section 2.8.2, we have at our disposal a matrix of best scores for each basephone at each frame in the utterance. This means that given a particular basephone at a particular frame we can calculate the overall rank of the basephone in question with respect to the other basephones. The predictor variable called "average senone rank" is calculated by averaging this value over the frames within the segmentation of each HYPothesized word. The distributions of average senone rank over the three word classes is given in Figure 6.10, which shows a slight separation between the (leftmost and tallest) class *correct* and the two other $not - correct$ classes.

6.7 Previous Word Values (ν)

Context, especially the left word context, can have a powerful effect on the creation of errors. If one word is guessed incorrectly a cascade effect can begin, sending the recognizer off to make additional errors. To attempt to capture this effect we add three predictor variables.

The first is simply the N-best homogeneity score of the previous (left-context) word. The idea is that if the recognizer was confused about the left context then there might be a higher chance of an error occurring with the current word.

The second and third predictors rely on the iterative use of confidence annotators. We build the best possible annotator without previous word context predictors, annotate each word using this "N-1" version annotator, and then use the $P(C)$ and $P(OOV)$ values of the previous (left-context) word as predictors for the "final round" annotator.

Results of this approach are reported in the next section.

6.8 Error-Predictive Power of Variables (ν)

Up to this point in the chapter we have enumerated the predictor variables we are working with and have taken a look visually at how much separation between the three word classes each gives us. Now we examine quantitatively how effective these variables are at predicting word errors. First, some shorthand for various subsets of predictor variables:

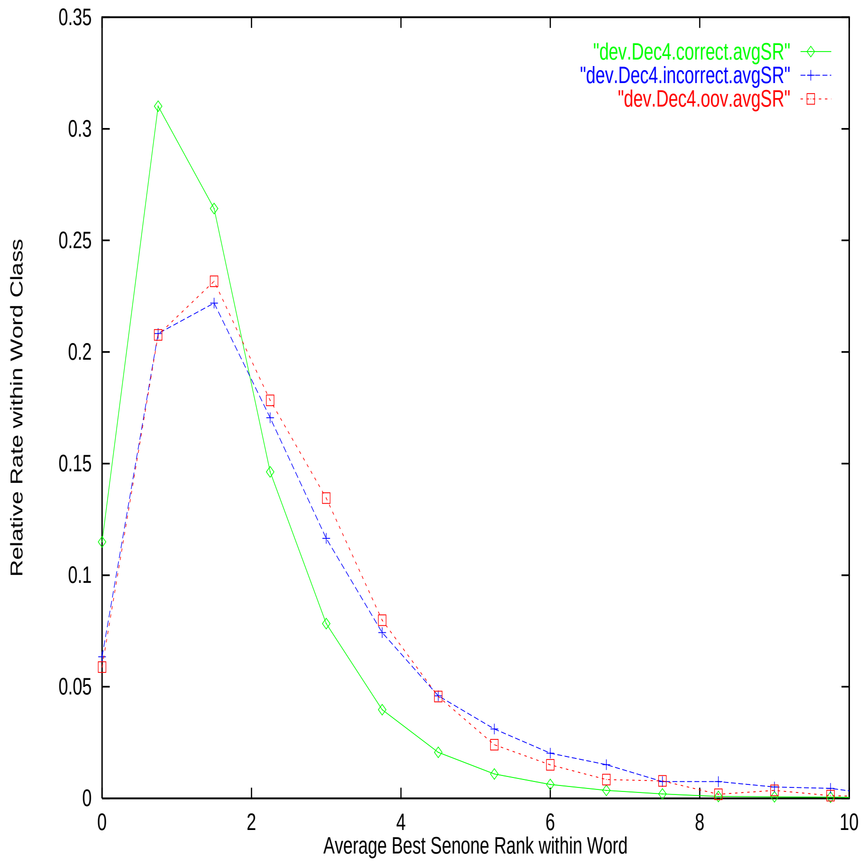


Figure 6.10: Average best senone rank for each class label in {*correct, incorrect/other, incorrect/oov*} from the NAB development data.

1. *BASE1:percPhAll*: The baseline system from Chapter 5 that uses the percentage of frames in the HYPothesized word whose basephones match the basephones in the phone-only decoding.

2. *BASE2:Combined+Duration*: The baseline system from Chapter 5 that uses the combined acoustic and language model score together with the duration (in frames) of the word.

3. *Nbest*: Only the N-best homogeneity score.

4. *Nbest+AvgWF*: The N-best homogeneity score together with the frame-averaged number of words present in the N-best list alongside the word in question.

5. *LMscore*: Language model score only.

6. *LMscore+LMsource*: Language model score together with the case/branch of its origin in the backoff algorithm.

7. *avgHPAll*: The average value of the H_{WC} metric of the word's phone from the phone-only decoding at the same point in the utterance.

8. *percPhAll+avgHPAll*: The combined simple baseline and H_{WC} metric.

9. *ConfBest+avgSenR+avgHPBest+acNormS*: A mix of several acoustic predictors that use the best senone at each frame as a reference. ConfBest is the result of applying the confusion-based distance metric, avgSenR is the average senone rank of the hypothesized phone, avgHPBest is the average H_{WC} value, and acNormS is the normalization by $P(A)_{SS}$.

10. *TrainWC*: Count of number of times the word was seen in the acoustic training data.

11. *Duration+Numphones*: The variables that relate to word duration: the duration in frames and the total number of word phones.

12. *prevPC*: Value of $P(C)$ applied by the almost-best annotator to the previous (left-context) word.

13. *prevPO*: Value of $P(OOV)$ applied by the almost-best annotator to the previous (left-context) word.

14. *prevNbest*: Value of the *Nbest* variable of the previous (left-context) word.

Each of these combinations of variables are used to produce a decision tree, using the procedure described for the baseline systems in Chapter 5. The decision tree is then used to produce confidence annotations, again as described previously. The trees are trained and the predictions are made across the three classes *correct*, *incorrect/other*, *incorrect/oov*. Thus, as in the case of the baseline, we can evaluate the overall performance in confidence annotation with respect to all three classes as well as the class-by-class performance. Table 6.2 summarizes the relative performance of each annotator.

Predictor Variables	%overall	%cor	%incor	%oov
BASE1:percPhAll	5.1%	6.3%	5.9%	2.5%
BASE2:Combined+Duration	5.3%	6.0%	4.5%	5.2%
Duration+Numphones	0.6%	0.3%	0.6%	1.2%
Nbest	8.7%	12.0%	8.1%	6.7%
Nbest+AvgWF	8.7%	12.0%	8.1%	6.7%
LMscore	2.0%	2.8%	1.2%	5.8%
LMscore+LMsource	2.5%	3.8%	1.6%	6.8%
avgHPAll	5.6%	7.1%	6.3%	3.1%
percPhAll+avgHPAll	5.7%	7.2%	6.5%	3.1%
ConfBest+avgSenR+avgHPBest+acNormS	3.8%	4.6%	3.3%	3.7%
acNormS	2.8%	3.2%	2.0%	2.9%
TrainWC	3.1%	2.6%	1.0%	5.5%
prevPC+prevPO+prevNbest	5.2%	5.7%	3.5%	4.3%

Table 6.2: Improvement in 3-class word confidence annotation as a function of the predictor variables used.

Most of these variables contribute to identifying the split between *correct* and $not - correct$ words or between *incorrect/oov* and $not - incorrect/oov$ words, but not both.

For example, language model score (LMscore) gives only a moderate amount of information about membership in class *correct*, but is the best predictor for class *incorrect/oov*. This explains why the BASE2 system, which uses combined word score (Combined) and word duration (Duration) as predictors, works better on class *incorrect/oov* than the BASE1 system: the combined score is correlated strongly with the language model score.

It is clear that the best predictor overall is the N-best list homogeneity score (Nbest). This is by far the best individual predictor for classes *correct* and *incorrect/other*, and performs almost as well as the best single predictor for class *incorrect/oov*. Nbest is such a strong predictor that when it is considered in conjunction with the average word frequency (AvgWF) predictor, it completely washes out any contribution from this, its less-effective relative.

All of the predictors (except N-best homogeneity) that give any noticeable contribution to the identification of class *incorrect/other* rely on comparisons between the HYPothesized phone sequences and the phone-only decoding. These are the BASE1 system (percPhAll), the H_{WC} metric applied to the phone-only decoding (avgHPAll), and the combination of the two (percPhAll+avgHPAll).

In comparing the H_{WC} phone/phone distance metric with the baseline BASE1 (percPhAll), which simply counted phone matches frame-by-frame, we see a slight improvement with the use of H_{WC}. This improvement is only evident for classes *correct* and *incorrect/oov*, however.

There are several predictors that do much better on class *incorrect/oov* than other classes: language model score (LMscore), language model source (LMsource), duration and numphones (Duration+Numphones), and word presence rate in acoustic training data (TrainWC). The predictors that "notice" differences in phone sequences between what was hypothesized and the phone-only decoding do not help much in identifying class *incorrect/oov*. Thus it seems that when the recognizer is forced to "cover" an unknown word in the input, it is likely to pick words that are acoustically close to, but not a perfect match for, the observed phone sequence, while backing off heavily in the language model. This produces a low language model score, using slightly longer words (by one phone on average), and often using words that were not trained explicitly from the acoustic training data.

The total contribution of the predictors based on the best-senone (and thus best basephone) at each frame appears to be quite small. There are four such predictors: the confusion-based phone/phone distance metric applied to the best senone per frame (ConfBest), the frame-averaged senone rank for the hypothesized phones in a word (avgSenR), the H_{WC} measure applied to the best senone per frame(avgHPBest), and the best-senone-score-normalized acoustic score (acNormS). The combination overall of these four gives a relatively passable performance on words in class *correct* but is relatively uninteresting otherwise.

This analysis of individual predictor variables does give us an idea of where we will get the best error-predictive power in our best system. But it is not the case that all of the predictors that seem promising here will be of use to us in the end, as nonlinearities in predictive power do exist. As we saw in the case of the two N-best list characteristic predictors (Nbest and AvgWF), it is always possible that one predictor will "wash out" the effects of one or more similarly motivated or similarly calculated variables. It is also possible that, due to specific interactions between predictors, in particular branches of a decision tree a variable that overall has very little impact might have a locally important contribution to make.

6.9 Performance of the Best Annotator (ν)

In this section we build and evaluate the best decision tree possible given the predictor variables we described in the previous section.

6.9.1 Domain-independent Evaluation (ν)

Because the recognizer uses gender-specific acoustic models, and because some of our predictor variables are tied very closely to the performance of the acoustic models, there is a question of whether we need to build gender-specific confidence annotators. *A priori* it would seem that our acoustically-related predictor variables are abstract enough with respect to the original acoustic scores that combining them before training the confidence annotator should probably work fine. But there is no real proof of this without an experimental check. Thus three different versions of this experiment are reported. The first two exploit gender-segregated versions of the development test data, which is the training data for the confidence annotator. The third version uses these two sets of data combined. Our hope in advance is that we will be able to combine the data without problems, as the total amount of training data for the confidence annotator is small.

We evaluate these three annotators in terms of reduction in cross-entropy, then move forward with a more detailed evaluation of the gender-combined system. We present the probability-of-correctness ($P(C)$) curves by word class on the independent evaluation test data to illustrate the amount of separation we can produce visually. We then sample these curves at various thresholds to indicate the range of tradeoffs possible in terms of false alarms and missed errors.

Confidence Annotator	% overall	% cor	% incor	% oov
BASE1:percPhAll	5.5%	6.9%	6.3%	2.8%
BASE2:Combined+Duration	5.3%	6.3%	5.4%	4.0%
Best	15.0%	19.0%	14.9%	11.7%

Table 6.3: Best male-only confidence annotation results compared to baselines. Performance measured in percent reduction in cross-entropy.

Confidence Annotator	% overall	% cor	% incor	% oov
BASE1:percPhAll	4.1%	5.0%	4.5%	2.4%
BASE2:Combined+Duration	4.9%	5.6%	3.3%	6.4%
Best	17.5%	21.4%	14.5%	17.0%

Table 6.4: Best female-only confidence annotation results compared to baselines. Performance measured in percent reduction in cross-entropy.

Tables 6.3 and 6.4 summarize the performance of the best combination of predictor variables on the respective gender-split portions of the independent test material. In both cases the variables referenced in the decision tree were the same: Nbest, LMscore, LMsource, percPhAll, acNormS, trainWC, avgWF, avgHPAll, Duration, Numphones, avgSenR, avgHPBest, and phConfBest. As we can see in Figures 6.11 and 6.12, in both cases the most important variables at the top of the trees were Nbest and LMscore.

The first split in these trees asks the question, "Is this word favored by most of the options in the N-best list?". For those words for which the answer is "Yes" the next question asked is, "Is this word very highly favored by most of the options in the N-best list?". Any word that passes both the first and second round of this question is extremely likely to be *correct*.

For those for whom the answer to the first question was "No", it is likely that the word is $not-correct$. But is it $not-correct$ because there was an OOV word spoken or is it $not-correct$ for some other reason? The next question asked is the most powerful splitter between class $incorrect/oov$ and

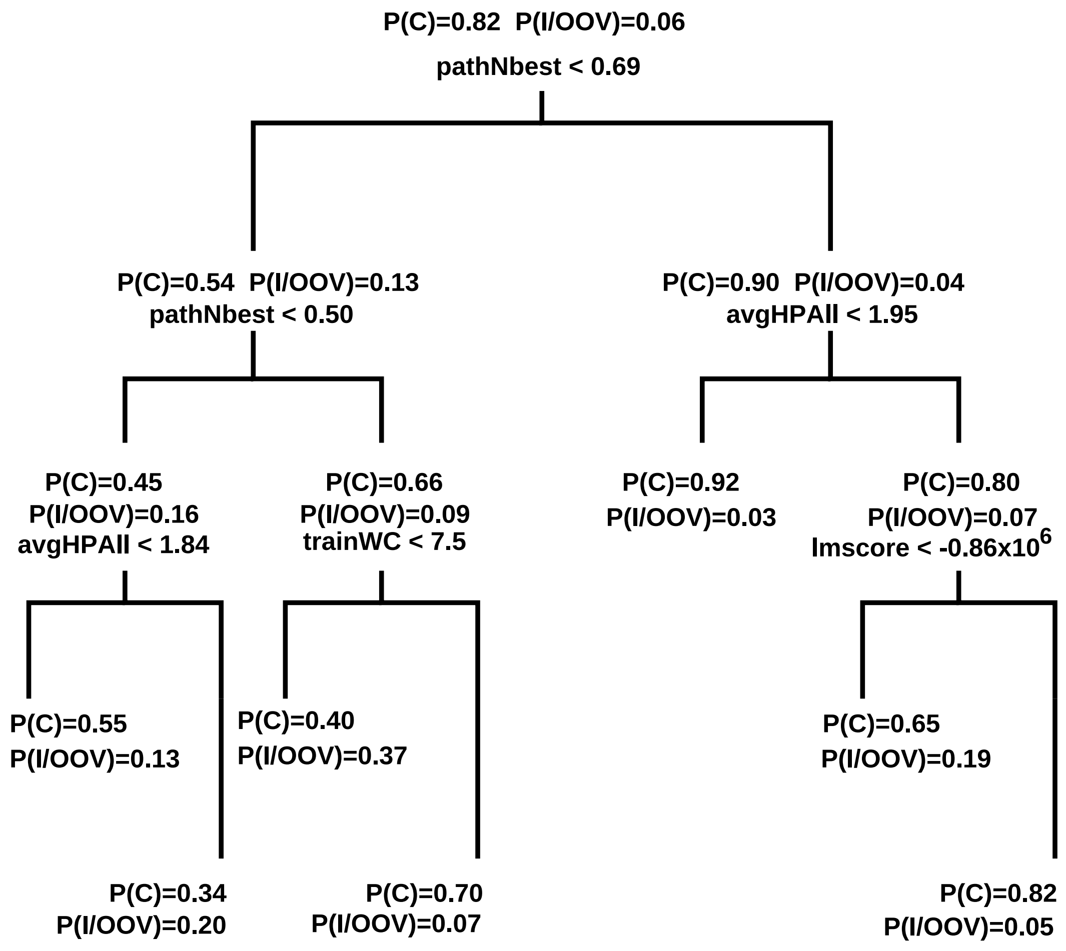


Figure 6.11: Top branches of the best male word-level predictor of error labels $\{correct, incorrect/other, incorrect/oov\}$.

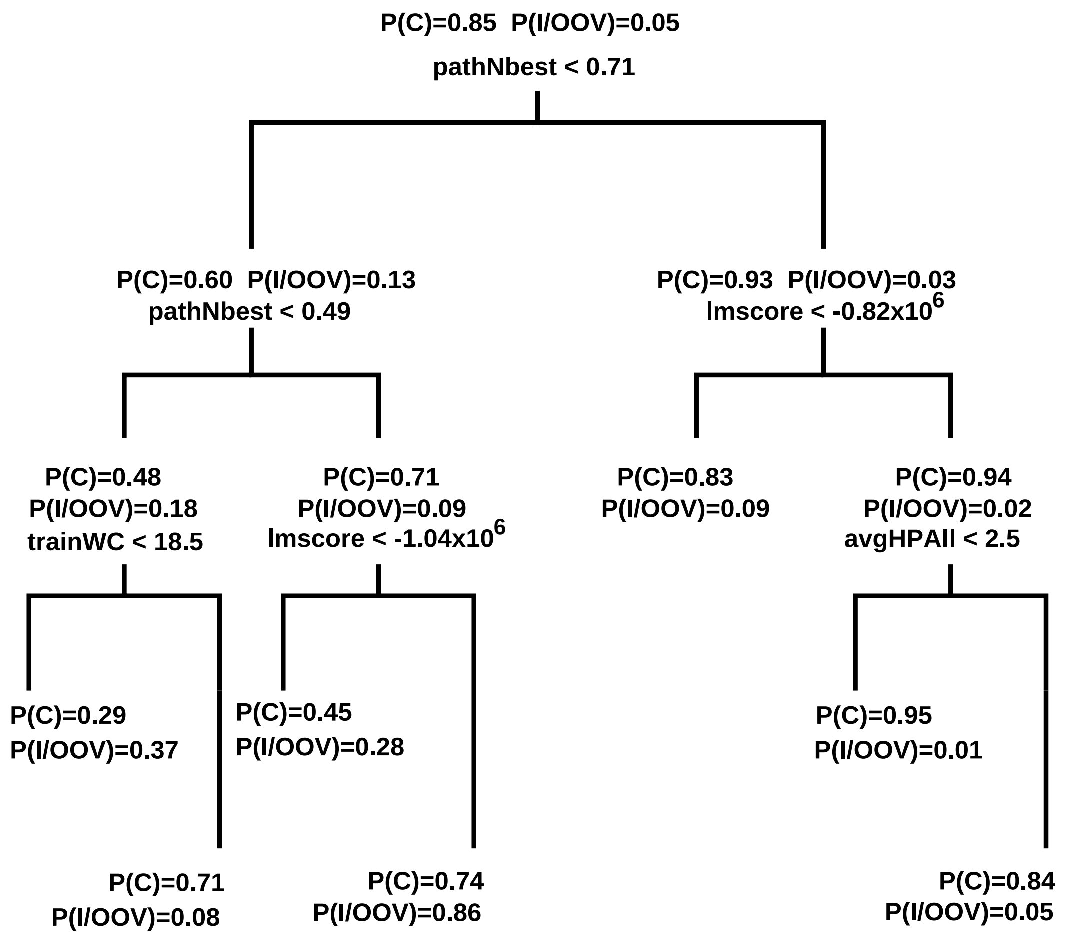


Figure 6.12: Top branches of the best female word-level predictor of error labels $\{correct, incorrect/other, incorrect/oov\}$.

the other classes – "Is the language score unusually low?". If the answer is "Yes", then the word is much more likely to be *incorrect/oov* than it was *a priori*.

In fact, as we can see in Figure 6.13, the same initial questions are the important ones for the gender-combined decision tree. Table 6.5 and Figure 6.14 show the best predictions made on the gender-combined data. In comparing this table with Tables 6.3 and 6.4, it appears that there is some sort of tradeoff going on between the benefits of having matched gender conditions between training and test (which are lost in the combined test) and the benefits of an increased amount of training data in the combined test. To see this, note that the performance on class *incorrect/other* is noticeably better in the combined test than in either the male or female tests alone. On the other hand, the performance on the combined test for both classes *correct* and *incorrect/oov* are between the male and female performance figures, indicating that some sort of compromise process is occurring.

Confidence Annotator	% overall	% cor	% incor	% oov
BASE1:percPhAll	5.1%	6.3%	5.9%	2.5%
BASE2:Combined+Duration	5.3%	6.0%	4.5%	5.2%
Best	17.5%	20.9%	16.5%	13.8%

Table 6.5: Best combined-gender confidence annotation results compared to baselines.

Consider the nature of a dictation system: in the error-correcting interface a user must specify the desired actual transcription somehow. We can rely on a simple piece of software to determine whether or not the word was known before the user entered it, thus mostly what we're trying to accomplish with confidence annotation in the context of a dictation system is the identification of errors. This means that we can use the results of our best-performing class, the identification of words in class *correct*, as the main focus of our evaluation. Since our combined system performs almost as well on the combined test data as the best (female) single-gender system does on its matched condition, we can push forward with the combined system for the rest of the detailed evaluation.

(We will return later in the chapter to look at the identification of OOV words, however, as this is in general an interesting problem. The identification of OOV words would be very useful in other task domains, such as

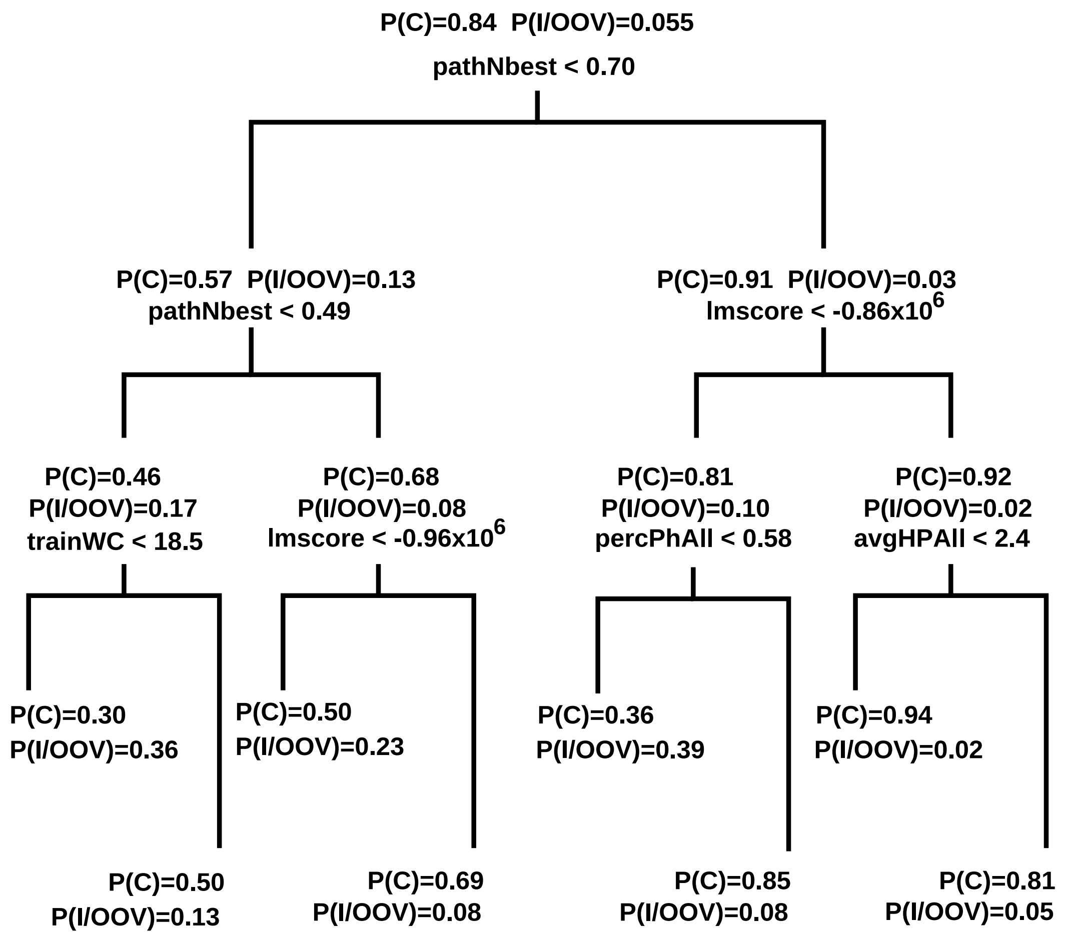


Figure 6.13: Top branches of the best combined gender word-level predictor of error labels $\{correct, incorrect/other, incorrect/oov\}$.

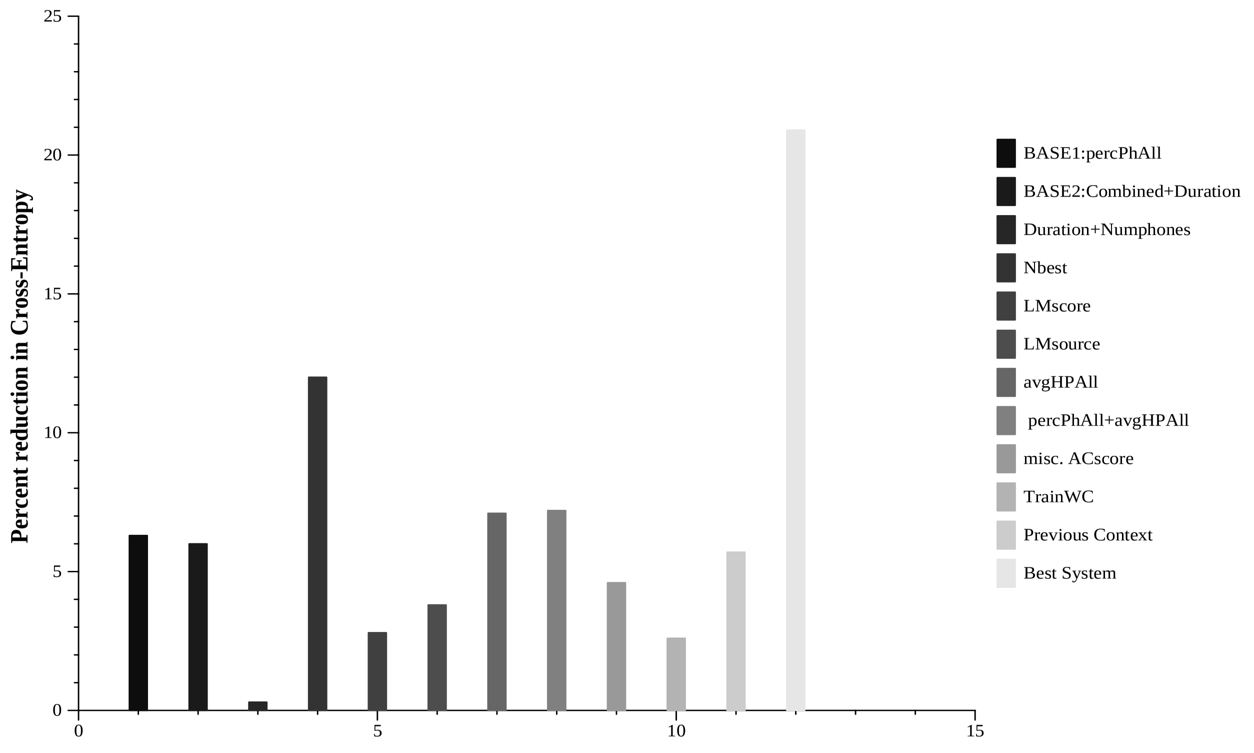


Figure 6.14: Incremental and best combined-gender confidence annotation results compared to baselines, NAB data.

spoken language interfaces with databases. In these systems if something is known about the nature of the error, rather than the blind fact that possibly an error has occurred, an appropriate clarification dialog can be initiated.)

Figure 6.15 shows the distribution of the probability of class *correct* applied by the best combined-gender confidence annotator to the independent test data. The x-axis of this figure is the $P(C)$ applied to the word by the annotator. The y-axis of this figure is the relative rate within a specific class at which a $P(C)$ on the x-axis was applied. The graph with a large peak at the right is the graph for class *correct*. The point $(0.8, 0.18)$ on this graph indicates that 18% of words actually in class *correct* were given a $P(C)$ annotation of between 0.8 and 0.9. This figure shows that:

- in general the confidence annotator applies high $P(C)$ values to words in class *correct* and lower $P(C)$ values to words in classes *incorrect/oov* and *incorrect/other*.

- the two *not−correct* classes receive similar distributions of $P(C)$ values, and

- the distribution of $P(C)$ for class *correct* is of a good shape to support eventual separation of the classes, but the distribution for classes *incorrect/oov* and *incorrect/other* has too much "mass" at the high end of the scale.

To compare with the *a priori* condition, consider the fact that initially each graph (one for each word class) was exactly the same. Each graph had exactly one peak at $(0.83, 1.0)$. This is because before the decision tree learning algorithm is applied each word has the same *a priori* chance of being in class *correct*. Thus one interpretation of the results in Figure 6.15 is that the confidence annotator shifts the bulk of the $P(C)$ applied to words actually in class *correct* to the right – more than 65% of these words are labeled with a $P(C)$ that is greater than 0.9 by the confidence annotator. The shift of $P(C)$ for the other two classes is in the right direction (to the left, lower values), but not far enough or fast enough to provide a perfect separation between the classes.

Two other good points of comparison are the same graphs for the two baseline systems described in Chapter 5. Figure 5.5 shows the same curves for the baseline system based on percentage of correct phone/phone guesses between the HYPothesized word and the phone-only decoding (percPhAll).

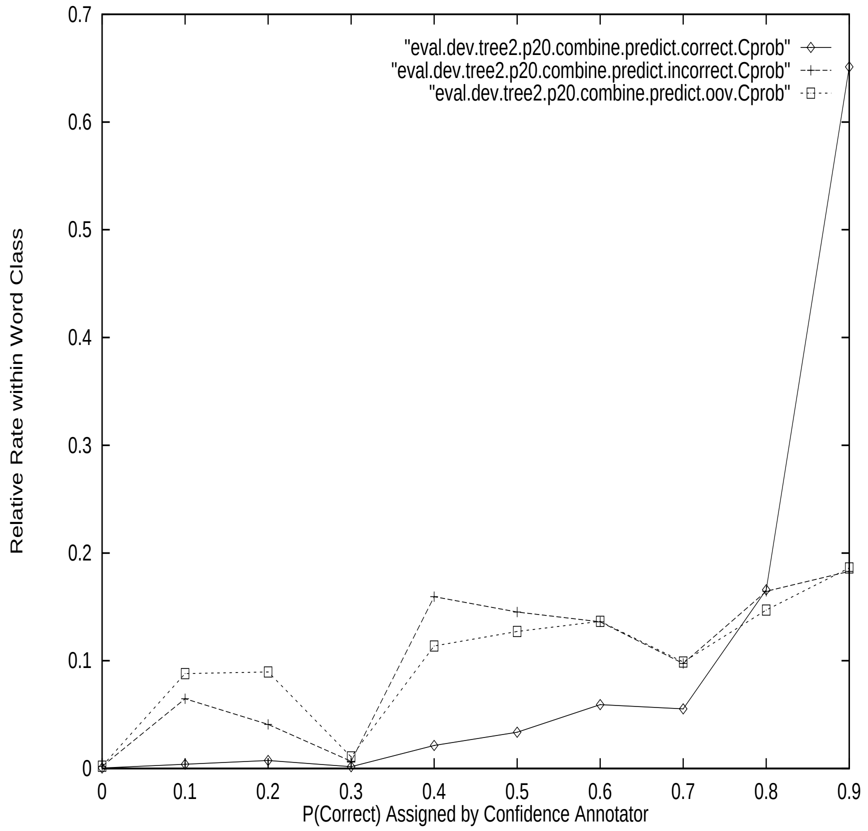


Figure 6.15: The $P(C)$ curves for the best word-level predictor of error labels in classes {*correct, not-correct*} in the NAB data.

Figure 5.6 shows the curves produced by the baseline system based on combined word score and word duration (Combined+Duration). The results in Figure 5.5 show a clearer visual separation between the *correct* and $not-correct$ classes, which is commensurate with that baseline's superior performance on class *correct*, as noted in Table 6.5. This system's curves also show that the shift to > 0.9 for $P(C)$ values on words in class *correct* is due in part to the percPhAll predictor variable on which the baseline relies.

It is possible (and generally preferable) to use probability annotations such as our confidence annotations in a smooth fashion, without imposing hard thresholds. But to analyze exactly how much separation the best system's $P(C)$ curves would supply in the event that we needed to use the confidence annotations to make a hard binary decision about class membership we sample these curves at various values of $P(C)$. This creates false alarm and missed error rates (of membership in class *correct*) that can be reported for each threshold.

To interpret Figures 6.17, 6.18, and 6.16, consider the following points:

1. The first figure in each pair refers to the rate at which the thresholded confidence annotator identified a hypothesized word as being $not-correct$, when in fact is was *correct*. The numerator of the first ratio (percentage) is this number of "false alarms". The denominator is the total number of words in the test set.

2. The second figure in each pair refers to the rate at which the hypothesized word was identified as being *correct*, when in fact it was $not-correct$. The numerator of this second ratio (percentage) is this number of "missed errors". The denominator is the total number of words in the test set.

3. There is a third figure related to these two that is not explicitly represented. This is the complementary percentage to the second – the rate at which hypothesized words are labeled accurately as being $not-correct$.

Figure 6.16 reports these rates for the best annotator's $P(C)$ curves (which appear in Figure 6.15). These values can be compared with those of the two baselines – the BASE1 figures appear in Figure 6.17, the BASE2 figures in Figure 6.18. Note that even though between the two baseline systems the BASE1(percPhAll) baseline performs best under the reduction in

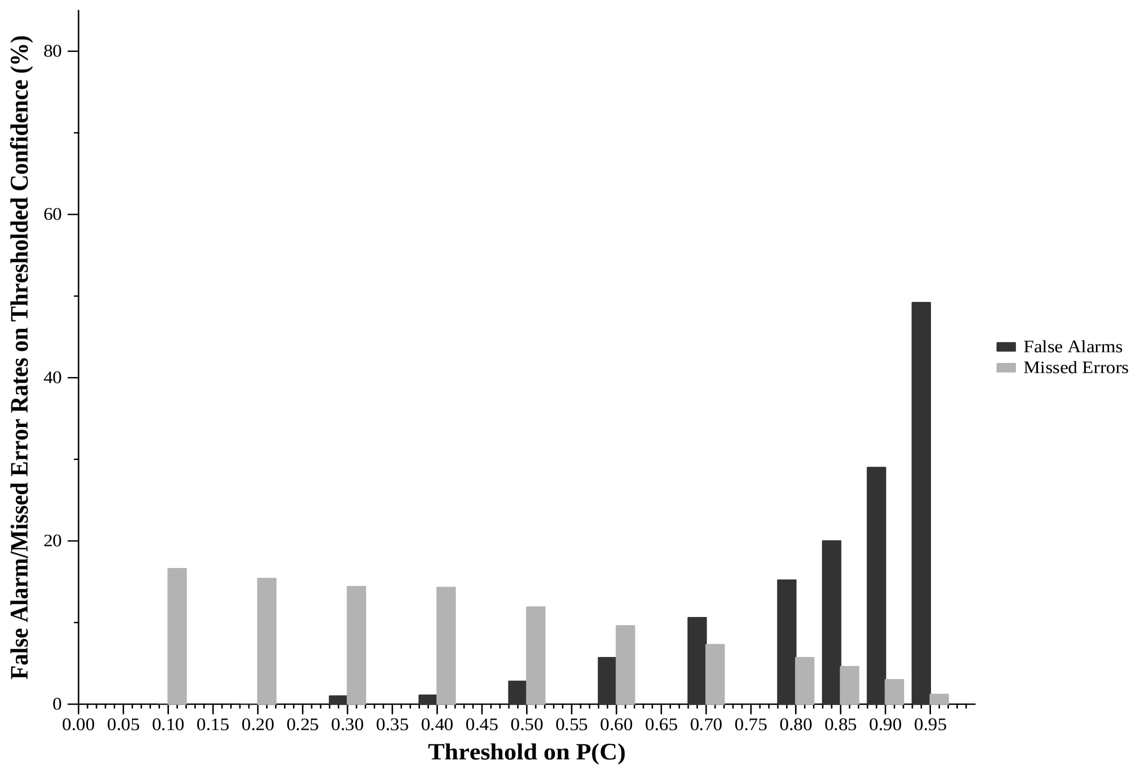


Figure 6.16: False alarm/missed error rates at various $P(C)$ thresholds for the best word-level predictor of error labels in classes {*correct,not-correct*}.

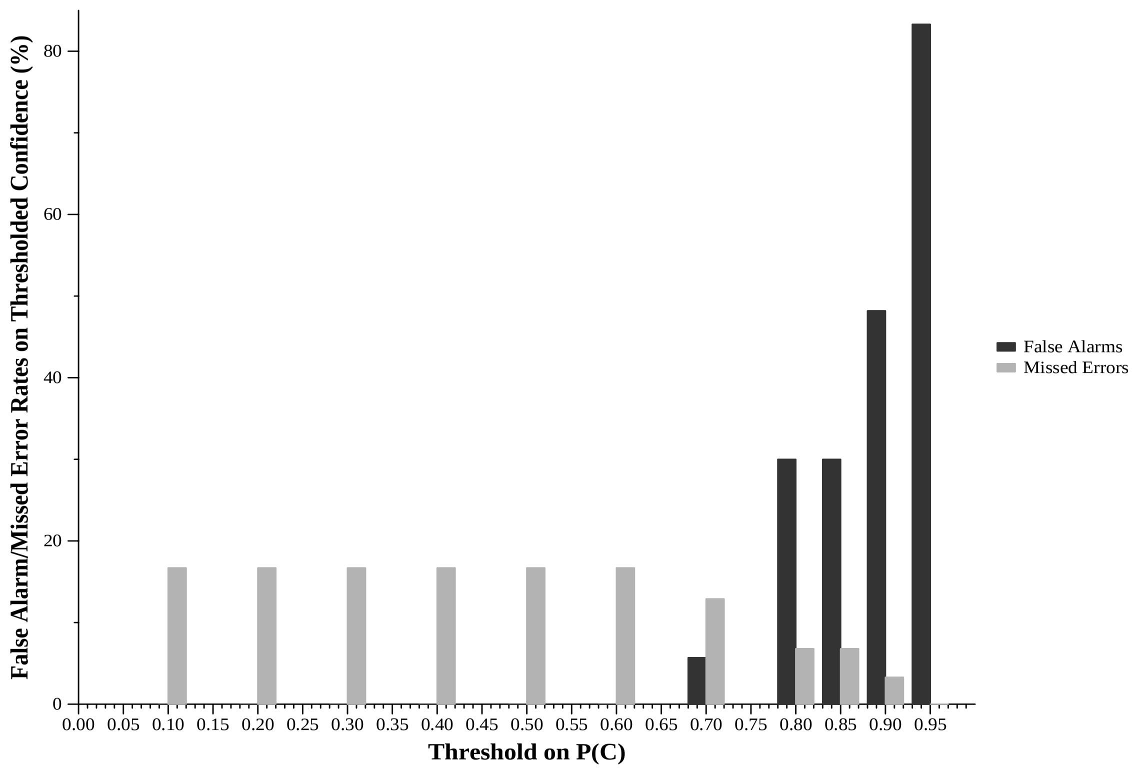


Figure 6.17: False alarm/missed error rates at various $P(C)$ thresholds for the BASE1 word-level predictor of error labels in classes $\{$*correct*,*not-correct*$\}$.

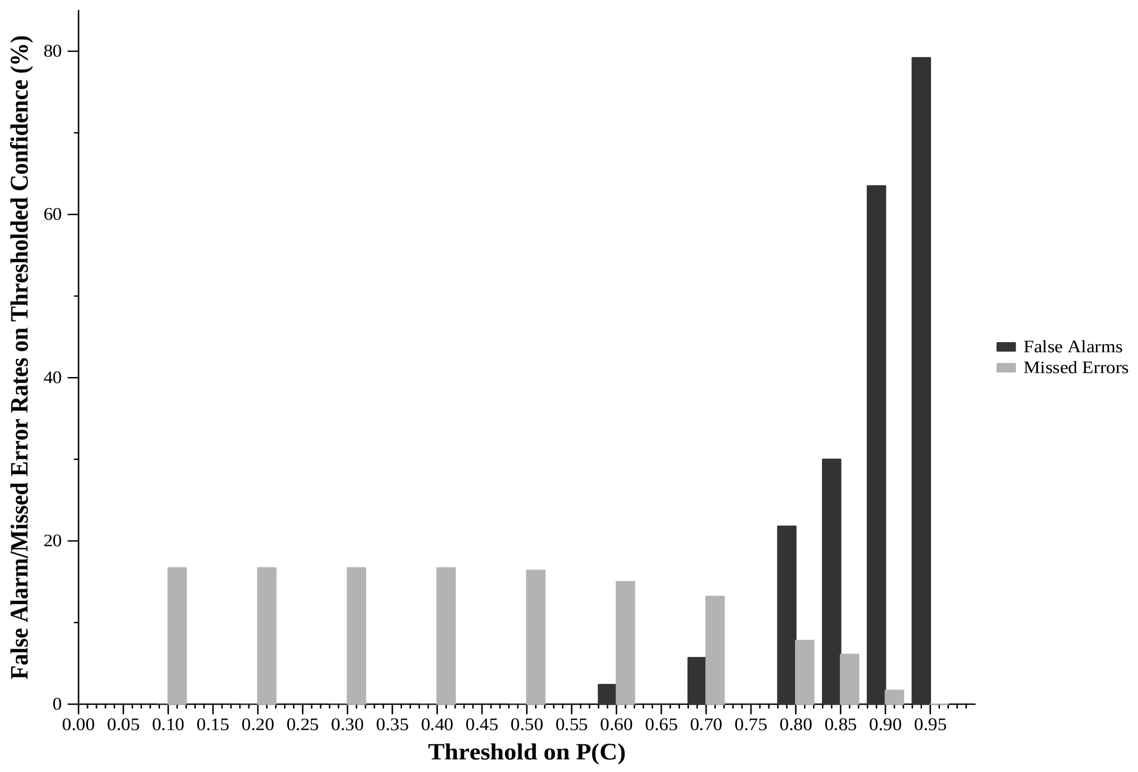


Figure 6.18: False alarm/missed error rates at various $P(C)$ thresholds for the BASE2 word-level predictor of error labels in classes $\{$*correct*,*not-correct*$\}$.

cross-entropy measure and the visual inspection of $P(C)$ curves, in this case it does not produce tradeoff points that are quite as useful. The BASE2 system provides slightly more room for choice in trading off alarms against extra checking labor.

In comparison with either baseline the best system of Figure 6.16 yields a much wider range of options in choosing a threshold on $P(C)$. In the abstract the results around the thresholds 0.7, 0.8, and 0.85 look pretty reasonable. But in order to evaluate the confidence annotator for use in any particular application, we need a model of the user's cost in performing activities related to each of the three points mentioned above. This evaluation may end up telling us where the best breakpoint on the $P(C)$ curve is, or it may end up telling us that the confidence annotator is simply not doing a good enough job to be useful.

6.9.2 Application-specific Evaluation (ν)

In this section we describe two examples of why application-specific measures are important for the evaluation of confidence annotators. First we will look at how confidence annotators can be used to filter regions of high quality recognition from regions of low quality output. This filtering has direct applications in many task areas, including tagging of multi-media data for retrieval.

Second, we will examine the use of confidence annotations in the context of an editing interface for a text-to-speech dictation application.

Using Confidence Annotators to Filter Speech Recognition In many applications of speech recognition it is useful to be able to identify regions of good quality output automatically. Consider, for example, use of a speech recognizer in the tagging of a multi-media database. In such cases the goal is to automatically create "searchable" reference tags for spoken language components of the database. These spoken language components might be radio broadcasts, which have no visual track. Or they might be television or film segments which have multiple parallel tracks. In either case, if a time-aligned transcript does not come ready-made with the material, an automatic speech recognizer can be used to create time-stamped labels. These labels can be used as the basis of retrieval searches on the stored segments.

Unfortunately automatic speech recognizers do not always perform well in "real-world" conditions in which speech must be recognized in very noisy

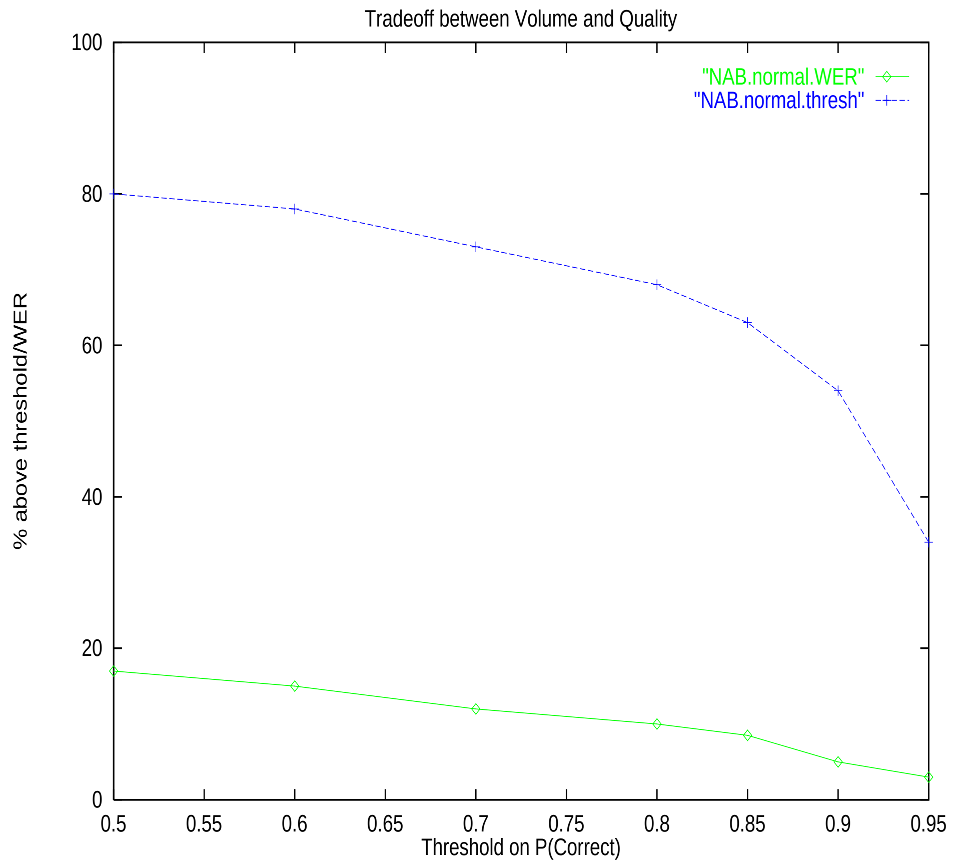


Figure 6.19: Tradeoff between amount of data passing filter vs. WER. NAB data.

P(C) Thresh	Avg WER	%age Above Thresh
0.5	17%	80%
0.6	15%	78%
0.7	12%	73%
0.8	10%	68%
0.85	8.5%	63%
0.9	5%	54%
0.95	3%	34%

Table 6.6: Word confidence annotations ($P(C)$ values) can be used to filter data for regions of high recognition quality (low WER).

or degraded conditions, or in over a background music sound track. Poor quality transcripts or time-marked tags on the speech elements of a database can impede retrieval capabilities substantially [21]. The use of confidence measures can increase the reliability of the tags applied to such database segments, at the cost of applying fewer labels per segment.

Table 6.6 shows this tradeoff between quality of tags and number of tags, demonstrated on the NAB data described earlier. Note that with a very high threshold on $P(C)$ we can achieve very good recognition performance, and still place 34% of the recognizer output as tags. Table 6.7 shows the complementary side of the filtering picture. In this table we summarize the ability to find regions of very low quality recognition.

An alternate view of this filtering behavior is presented in Figure 8.23. The upper curve in the figure represents the amount of data passing through the filter. The lower curve represents the quality of the data that passes through. We can see from this figure that at lower threshold values on $P(Correct)$ the tradeoff between these two is relatively even, but that at higher thresholds the amount of data that passes the filter drops much more sharply in comparison with gains made in the quality of the filtered data.

This sort of filtering has direct application in improving retrieval from multi-media databases. If a speech recognizer is used to automatically tag speech tracks in such databases, confidence annotation can be used to improve access performance. In this sort of application, it is important to identify "content" words that relate directly to the subject matter of the items to be retrieved with the highest possible accuracy. Table 6.8 and Fig-

P(C) Thresh	Avg WER	%age Below Thresh
0.3	78%	17%
0.4	76%	18%
0.5	62%	20%
0.6	54%	22%
0.7	44%	27%
0.8	37%	32%
0.85	33%	37%
0.9	26%	48%
0.95	18%	66%

Table 6.7: Word confidence annotations ($P(C)$ values) can be used to filter data for regions of low recognition quality (high WER).

ure 6.20 demonstrate the filtering capabilities of the confidence annotator on the content words of the NAB data.[6] The confidence annotator, which was trained on both content and non-content words, does a better job overall of identifying content words than it does in the general case. Figure 6.20, which shows the filtering tradeoff between number of content words passed and recognizer accuracy of content words, indicates this performance by having both curves on the chart in more extreme positions than in the baseline Figure 8.23. For example, at a threshold value of 0.8, 73% of all content words pass the filter, while only 68% of words overall make it through. At this level of filtering, the word error rate in general is 10%, while the word error rate on content words is only 7.7%.

Confidence Annotations on Dictations In order to demonstrate the possible use of confidence annotations in the editing interface of a text-to-speech dictation system, we will present a simple example. Let's assume that the results of the confidence annotator will be used in the following way:

- The computer on which the dictation system is running has a screen, a keypad, a mouse, and a microphone.

[6]A content word is defined to be any word not in the top 100 frequency words in the vocabulary. Content words cover about half of the experienced words in both the development and independent test sets.

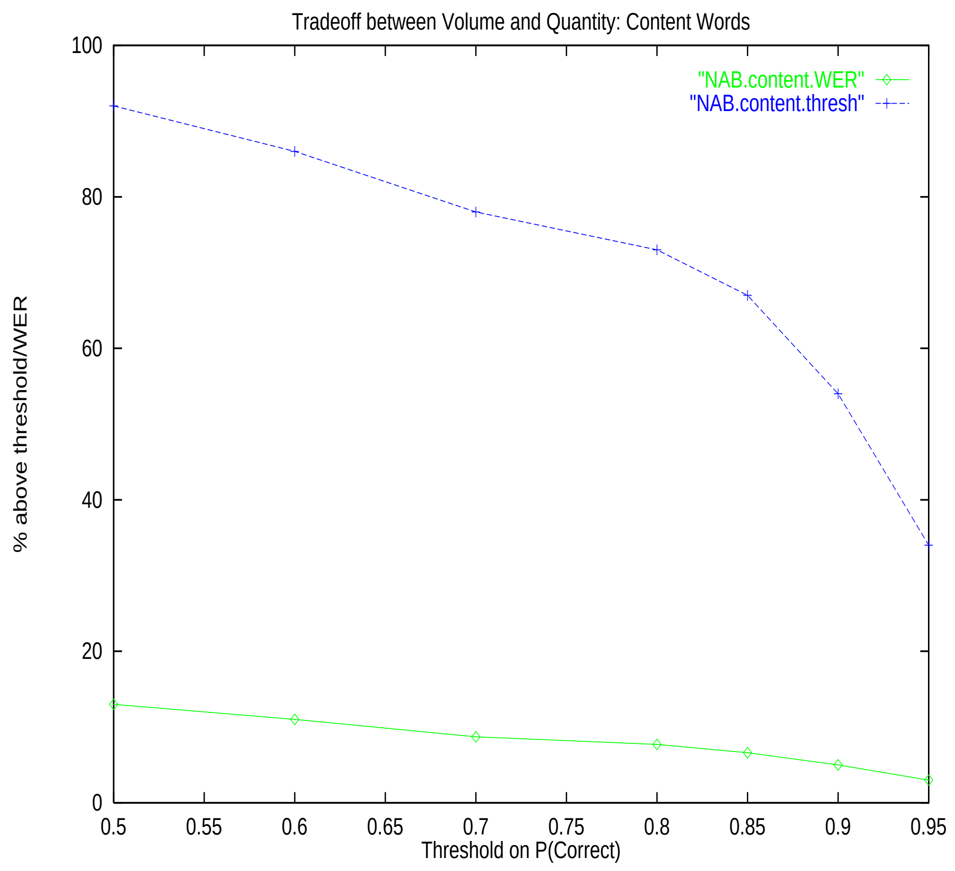


Figure 6.20: Tradeoff between amount of data passing filter vs. WER. Content words in NAB data.

P(C) Thresh	Avg WER	%age Above Thresh
0.5	13%	92%
0.6	11%	86%
0.7	8.7%	78%
0.8	7.7%	73%
0.85	6.6%	67%
0.9	5%	54%
0.95	3%	34%

Table 6.8: Filtering for "content" words in the NAB data set.

- The dictation system sits on the desk of the user, placing the dictated words on a screen in front of the user in roughly real time as the user speaks.
- The confidence annotator decides "behind the scenes" whether each word is *correct* or $not - correct$. Those words labeled as *correct* are placed on the screen in blue. Those labeled as $not - correct$ are placed on the screen in red.
- The user of the system ignores all supposedly *correct* (blue) words, paying attention only to those that appear to be wrong (red).[7]
- If the user of the system thinks that a red word is actually *correct*, he or she can simply ignore it. (In effect, the system is saying, "Hey! I'm wrong!" and the user is tacitly saying, "No, You're not.")
- If the user agrees that a red word is actually $not - correct$, he or she double clicks on it and a correction window pops up, into which the user types the correct word.[8] (In this case the system is saying, "Hey! I'm wrong!" and the user is actively answering, "Yes!, You are, and here's how to be right.")

[7]This is one highly unrealistic part of our simple example. At the very least people will generally notice that the confidence annotator has missed errors in blue words that are near red ones!

[8]Here's where the dictation software can decide whether this was an OOV error or not and deal with it accordingly.

Under this model any word that is actually $not - correct$ but is labeled as $correct$ will slip past, leaving an error in the final dictation produced by the system.

In order to evaluate how well the confidence annotator is working in this model, we need to estimate costs for three aspects of the model:

1. How much effort does the user expend in ignoring a red word?
2. How much effort does the user expend in agreeing that a red word should be red and then giving a correction?
3. How much does the user care about errors that slip past unnoticed?

If it were three times harder to type in a correction than to click on a word, and if missing an error cost twice as much as typing in a correction, then the relative cost of items 1, 2, and 3 could be set at 0.1, 0.3, and 0.6 respectively. Under these conditions, the best threshold in from Figure 6.16 will be between 0.6 and 0.7. If, however, errors that slip past don't matter much to the user, then we might use the respective costs 0.2, 0.6, and 0.2. In this case the best threshold would be 0.9 or higher.

If in addition we allow the consideration of constraints such as, "I don't want more than 5% of the total words produced in the dictation to be wrong," then we can use a set of these costs to estimate how much total work the user will have to do given the confidence annotator in question.

For example, if we use a 5% limit on total errors that slip through, then the threshold we would select would be 0.85, which allows 4.6% error slippage. If we estimate that typing in a correction of a red word is four times more expensive than ignoring it, then we use the relative costs of 0.8 and 0.2 for the actions. At the confidence threshold we've chosen the user will ignore 20 red words and do 12 word corrections for every word he or she dictates. If it takes one second on average to ignore a red word then it will take four seconds to type in a correction. The user will thus spend 68 seconds, or a little more than a minute correcting errors for 100 words' worth of speech.

In the NAB data speakers take about 40 seconds to produce 100 words of speech. Thus, in terms of time spent by the user, the repair effort under this model (allowing up to 5% error slippage) is approximately 2/3 of the total effort of dictation. This figure can be compared with the cost of editing non-annotated recognizer output by hand. For instance, let's assume that a user of our dictation system can check the recognizer output for accuracy by hand with 5% error slippage at a rate of two words per second. It will still take four seconds to type in corrections for wrong words, so the user will spend 50 seconds checking and 48 seconds typing corrections, for a total of 98 seconds of repair work for 100 words of speech. Under this cost model we've apparently shaved 30 seconds off of the correction time by using the confidence annotator.[9]

General Comments These examples are not meant to be exhaustive tests of the efficacy of this particular confidence annotator. Rather, this discussion has been intended as detailed motivation for the use of domain-specific evaluation of confidence annotators in general. By looking at these examples we have seen that domain-specific evaluation of confidence annotations is useful for both

- comparing confidence annotation thresholds against each other, and
- deciding whether the confidence annotator should be included in the recognition system at all.

It could be that some combined recognition/annotation systems would actually perform better in conjunction with some dictation interfaces without annotation if the confidence annotator were not accurate enough. For instance, if the medium of the dictation system is a voice-only telephone channel, the demands on the performance of the confidence annotator will likely be much higher than in the case described here.

It could also be the case that two separate confidence annotation schemes might look similarly powerful under the cross-entropy reduction measure, but that in the context of one particular application one of them might perform better. (See Section 8.3.3 for an example of this.)

In any case, it is not enough to simply look at reduction in cross-entropy or $P(C)$ curves when choosing a confidence annotator for an application system. Domain-specific measures definitely give more insight into the eventual performance of confidence-annotated recognizers than do more general methods.

[9]This example, not tied to any experimental work directly, is presented simply as an example of how cost models of false alarm/missed error tradeoffs should be performed.

6.10 Identifying OOV Words (ν)

We can look for OOV words in the spoken input by examining the $P(OOV)$ curves in much the same manner as we did the $P(C)$ curves above. Figure 6.21 shows the $P(OOV)$ curves produced by the best confidence annotator. These can be compared with the baseline results from the BASE2 system, which had the better performance on class *incorrect/oov* between the two baselines. The baseline curves appear in Figure 6.22. Note that between the baseline and the best system the distribution of $P(OOV)$ has changed very little for words in class *correct*. The greatest changes occur for words in class *incorrect/oov*, which shows some hopeful signs of spreading in the correct direction (higher values of $P(OOV)$), but which does not appear to be very well modeled.

The corresponding false alarm/missed OOV values for these curves are presented in Figure 6.23. Setting the threshold at 0.05 on the $P(OOV)$ curve means that 70% of the OOVs in the data are found. But to get this rate of identification we have to pay a cost of dealing with false alarms – 25% of the words not in class *incorrect/oov* are wrongly identified as being in class *incorrect/oov*. This means that in 100 words of dictation that includes 5 OOV words, we will find 3 or 4 of the OOVs, but we will have to deal with 24 false alarms. This is not very good performance. Most of the false alarms are actually correctly-decoded words. Thus in our example 17 of the 24 false alarms would be words actually in class *correct*, with the remaining 7 words identified as covering OOVs actually being errors due to some other cause. Table 6.9 contains a summary of similar results calculated for a range of thresholds on $P(OOV)$.

$P(OOV)$	#OOVs found	#FA on correct words	#FA on errors
0.05	3.5	17	7
0.1	2.8	9	5
0.15	1.9	6	2
0.2	1.4	3	1
0.25	0.9	1	1

Table 6.9: Performance on finding 5 OOV words in 100 words of dictation for the best confidence annotator at various thresholds of $P(OOV)$.

These overall results in predicting OOVs do not compare favorably with

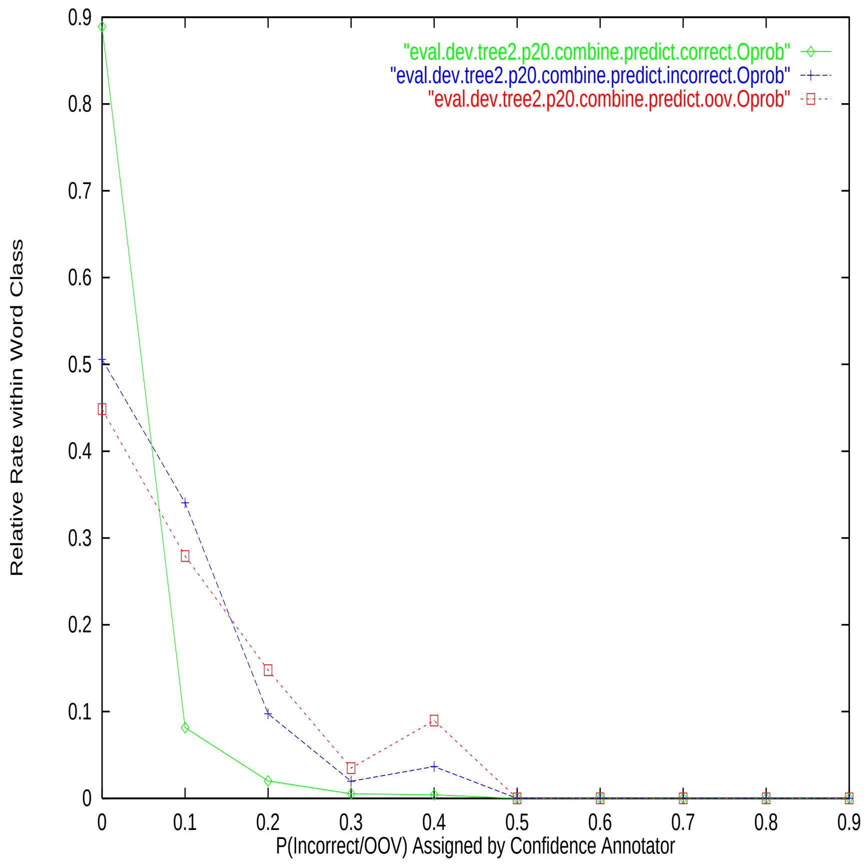


Figure 6.21: The probability-of-OOV curves for the best word-level predictor of error labels in classes {*correct, incorrect/other, incorrect/oov*}.

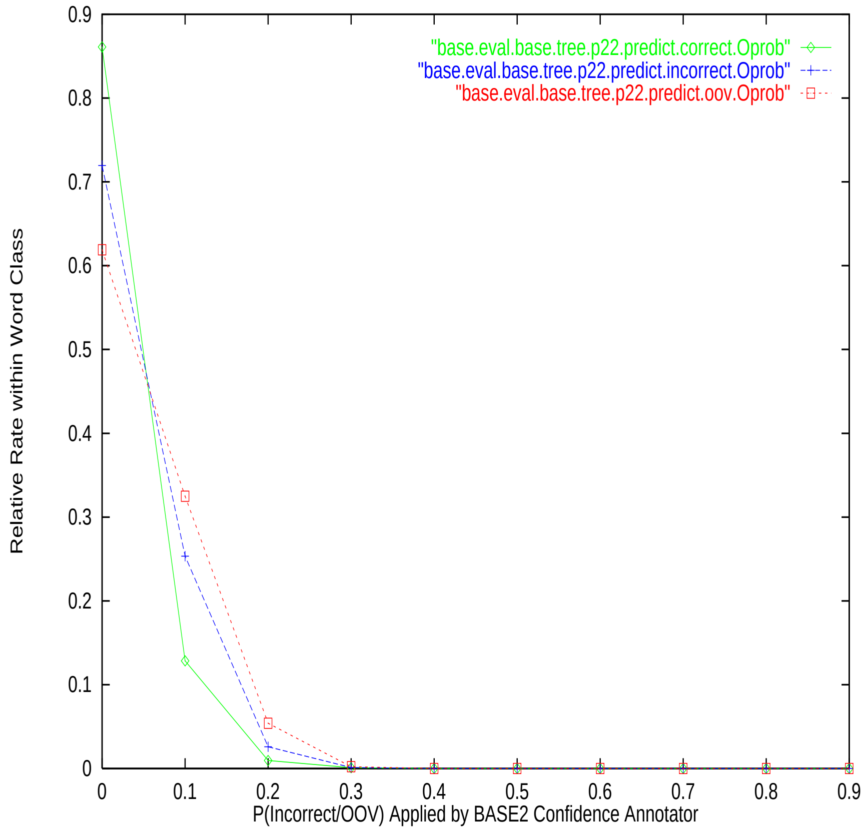


Figure 6.22: The probability-of-OOV curves for the BASE2 baseline predictor of error labels in classes $\{correct,\ incorrect/other,\ incorrect/oov\}$.

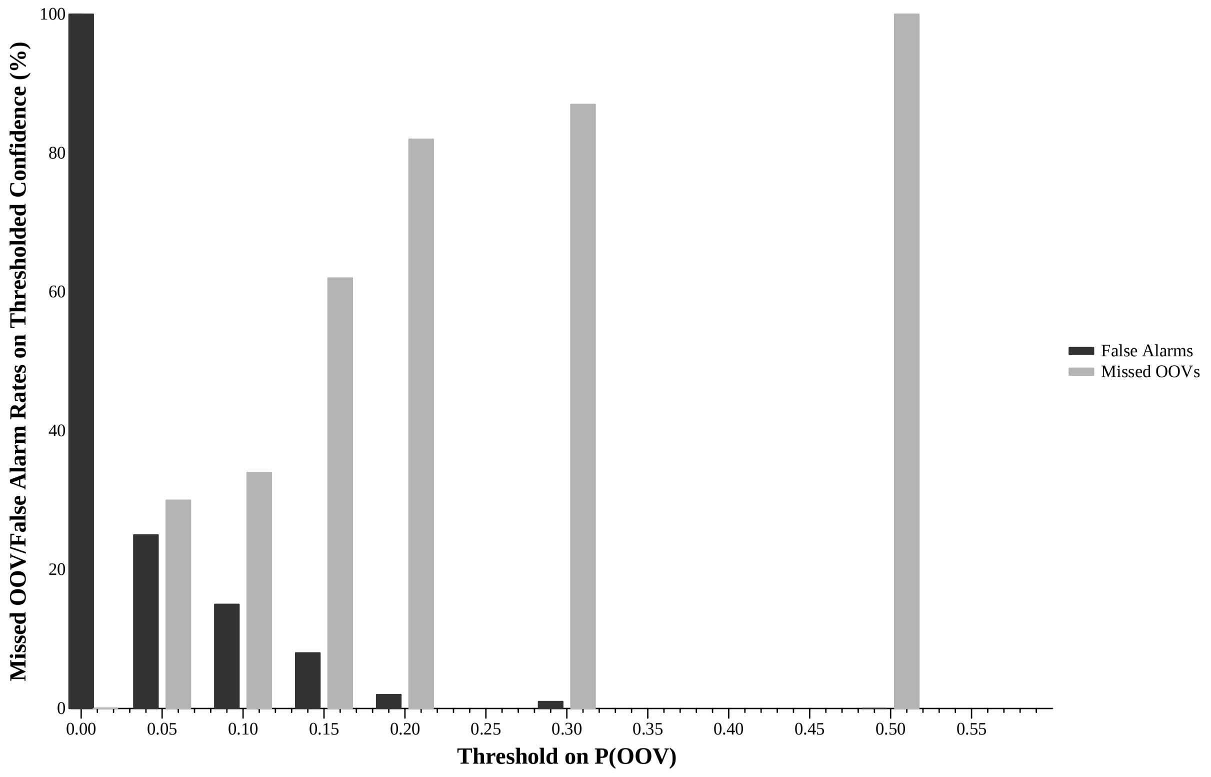


Figure 6.23: False alarm/missed OOV rates at various $P(OOV)$ thresholds for the best word-level predictor of error labels in classes *incorrect/oov*, *not−incorrect/oov*.

previous best attempts at the task, such as in Asadi's Ph.D. thesis [2]. It should be noted that the task in which we are attempting to identify OOVs is a much harder recognition problem overall than was Asadi's task. Also, in his domain the rate of OOVs was generally larger than the rate of errors due to other reasons. This is not true in our case. The approach reported here is extremely general and does not require any system-specific tuning of search parameters or models, which was the central technique in Asadi's work. Thus overall this approach cannot be said to have solved the OOV identification problem, but it is an interesting and potentially more general approach to a harder version of the problem.

6.11 Summary

In this chapter we developed a method of applying confidence annotations to the word-level output of a speech-to-text system. We also demonstrated a variety of evaluation metrics in detail. Our best confidence annotator separates correctly recognized words from those not correctly recognized approximately four times better than two simple baseline systems, with an overall reduction in cross-entropy for the labeling of class *correct* of 20.9%.

In the experiment reported here, the most powerful predictor variable that contributes to this best system's identification of correct and not correct words is called the "N-best homogeneity score", or "pathNbest". It is a measure of how many options the recognizer is considering at any point in time during the utterance – regions in which the recognizer is considering only one or two options are generally correct. For those words whose values of pathNbest are lower than about 70%, the next most important factor in deciding the likeliness of correctness is again pathNbest. In effect our best confidence annotator asks first, "How sure is the recognizer that this is the word?" For those words for which the answer is "Not really sure, there is some confusion." the next question is, "OK, there's confusion. How much confusion?"

For those words for which the value of pathNbest indicates little or no confusion, however, the next most important predictor variable is language model score. The second question, "OK, there's not much confusion. How good is the language model score?", indicates primarily whether there is an OOV present in the input. For those words for which the answer is, "The language model score is very low," the likelihood of the word being in error due to the presence of an OOV word in the input is very high. In fact, the best overall predictor variable for finding OOV words is the language model score. As we saw in Chapter 4, the recognizer tends to "cover" OOV words by picking words that are acoustically similar to the spoken OOV word. In order to "stretch" to find reasonable matches it is often necessary to insert words that have low language model scores. This often is accomplished by the use of the most extreme branch of the backoff algorithm – backing off to unigram scores is much more common for the decoding of OOV words than it is for errors of other types and correct words.

After these initial tests, the next important tests are comparisons with the phone-only decoding. If there seems to be little confusion in the N-best list but the all-phone decoding does not match the HYPothesized word well, then the probability of error increases dramatically. Two of the three distance metrics examined for comparing the HYPothesized phones with the phone-only decoding are useful in this role – the percent match of phones at the frame level and the H_{WC} phonological similarity measure.

Another important test near the top of the tree is the number of times the word being HYPothesized was seen in the acoustic training material. In cases that appear to be errorful because of high apparent levels of confusion, this predictor can distinguish well between errors due to OOVs and those due to other problems.

Overall performance in OOV detection is not as good overall detection of errors. The best OOV annotator achieves only 13.8% reduction in cross-entropy for this class, as opposed to the almost 21% for the main problem. These results in predicting OOVs do not compare favorably with previous best attempts at the task, such as in Asadi's Ph.D. thesis [2]. However, the approach reported here is extremely general and does not require any system-specific tuning of search parameters or models, which was the central technique in Asadi's work. In comparison the technique used here can be considered a "poor man's" approach, as it demands no work in addition to the confidence annotation requirements.

The main successful point of doing the three-class confidence annotation in this experiments is that in trying to predict OOVs we actually improve our performance in predicting all errors. Using exactly the same set of predictor variables in a two-way prediction problem in which classes *incorrect/oov* and *incorrect/other* were collapsed into one, performance suffered by almost 2% (absolute) cross-entropy reduction.

In addition to the cross-entropy reduction measure we look at differences in probability distribution curves to gain insight into the performance of our best annotator. By thresholding these curves at various points we can also produce false alarm and missed error rate pairs for the identification of correct words. These methods are provided as alternative perspectives for the domain-independent evaluation of confidence annotator performance.

We also illustrate the importance of the use of task-specific performance measures by looking at how confidence annotation can be to filter recognizer output for high and low quality regions. We discussed a potential application of such a filtering scheme in the improved tagging of speech tracks in multi-media databases. We also described a simple example of how confidence annotations might be used in the editing interface of a text-to-speech dictation system.

Chapter 7

Phone Confidence Annotation

The experiments described in the previous chapter focused on predicting how well the entire recognizer (the acoustic and language model scoring units combined) performed in making correct hypotheses. In this chapter we focus instead on how to predict whether the acoustic scoring mechanisms alone are working at any point in time. To do this, the level of detail at which we train and make confidence annotations is shifted from the word level down to the phone level. This produces an annotation mechanism whose labels can be interpreted as *acoustic confidence annotations*.

In most cases the predictor variables used are simply the same acoustic predictors used in the previous chapter, but scaled down to the phone level. Two additional predictors are added, as described below.

The predictor variables are constructed from a combination of recognizer-internal measures and additional parallel computations, including:

- Phone segmentations and acoustic scores from the best-scoring HYPothesis produced by the recognizer,

- Characteristics about the words in which the phones appeared when HYPothesized, including their pronunciations and how frequently they appear in acoustic training materials,

- The results of a parallel "phone-only" decoding, in which the recognizer is not constrained to either phone sequences from the dictionary or word sequences in the language model in its recognition of phone sequences,

- The results of a completely unconstrained frame-by-frame decoding in which the best possible acoustic score and basephone are determined for each frame of the utterance,

- Three distance metrics between basephones, including a simple match count at the frame level, a phonologically-based similarity measure (H_{WC}), and an empirically derived confusion-based distance measure, and
- The contents of an N-best list that contains complete phone segmentation and score information for each of 150 elements.

After constructing and analyzing the error-predictive power of each variable in isolation, we compare various combinations in their ability to work together in predicting errorful phones. A variety of subgroupings are compared using the reduction in cross-entropy measure developed in Chapter 5. The collection of predictor variables that performs best under this metric are then evaluated, with the performance results reported in several ways:

- overall reduction in cross-entropy of labelings,
- visual comparison of probability distributions for membership in phone class *correct*,
- a comparison of false alarm/missed error rates at various threshold points on the probability-of-correctness curve.

7.1 Defining Phone Correctness (ν)

For the purposes of this experiment we will attempt to predict whether phones guessed by the recognizer in a HYPothesized word sequence are *correct* or *incorrect*. We will not concern ourselves with trying to distinguish OOV-related errors from other errors using the new acoustic-only confidence annotator. The results from the previous chapter show that:

1. in general the language-based predictors are the most useful in separating OOV-caused errors from others, and
2. the two classes of errors were essentially indistinguishable with respect to acoustic predictors.

In order to set up the training for our phone confidence annotator we must first analyze the development test data, labeling each phone segmentation in the HYPothesis sequences as either matching or not matching their counterparts in the REFerence sequences. We do this exactly as described in Section 4.3, using a tolerance of two frames. The data used is the NAB data set; the recognizer is Sphinx-II. All of this is directly analagous to the work reported in Chapter 6.

7.2 Predictor Variables Used (ν)

All of the appropriate acoustic predictor variables explored in the previous chapter are used. Some of these have been extended or transformed slightly in order to fit the new phone-level requirements. Each variable used is describe in turn below, and each is presented together with a figure that indicates graphically how much separation is supplies between the phone classes *correct* and *incorrect*. If the predictor variable used here has an analagous counterpart in the previous chapter, we note the name of the predictor in parentheses next to its current description. As in the previous chapter a short name token is associated with each predictor. This tag will be used as a reference to the predictor variable in tables and discussion.

- *Numphones*: The number of basephones in the dictionary pronunciation of the HYPothesized word in which the phone in question was found. See Figure 7.1. (Numphones).

- *PhonePos*: The position of the phone in question in the dictionary pronunciation of the HYPothesized word in which it was found. See Figure 7.2. (None).

- *TrainTPC*: The number of times the triphone in question appeared in the acoustic training data. See Figure 7.3. (TrainWC).

- *TrainBPC*: The number of times the basephone in question appeared in the acoustic training data. (TrainWC).

- *NbestPhone*: The percentage of paths in the N-best list that pass through the basephone in question, weighted by N-best overall path score. See Figure 7.4. (Nbest).

- *acNormSphone*: The phone acoustic score normalized by the best senone score possible for the same frame sequence. See Figure 7.5. (acNormS).

- *BASE:PhAll*: The percentage of frames that match between the phone in question and the phone-only decoding. See Figure 7.6. (BASE1:percPhAll).

- *PhBest*: The percentage of frames that match between the phone in question and the best-senone basephone. See Figure 7.7. (percPhBest).

- *HPAll*: The H_{WC} distance between the phone in question and the phone-only decoding. See Figure 7.8. (avgHPAll).

- *ConfAllPhone*: The confusion-based basephone distance metric applied to the phone in question and the phone-only decoding. See Figure 7.9. (ConfAll).

- *HPBest*: The H_{WC} distance between the phone in question and the best-senone basephone at each frame. See Figure 7.10. (avgHPBest).

- *ConfBestPhone*: The confusion-based basephone distance metric applied to the phone in question and the best-senone basephone at each frame. (ConfBest).

- *avgSenRPhone*: The frame-averaged rank of the basephone in question according to the senone scores at each frame. See Figure 7.11. (avgSenR).

- *Duration*: Phone duration in frames. See Figure 7.12. (Duration).

- *ACScore*: Acoustic score of the phone. (None).

Two new predictors have been added for these experiments. The purpose of these predictors is to measure the characteristics of the spread of phones found in the N-best list in the temporal vicinity of the phone being analyzed. The two new predictors are essentially refinements to the NbestPhone predictor described above. Instead of simply measure *whether or not* the N-best list is homogeneous at some point, they each try to measure something about *how* the N-best list is inhomogeneous.

The extensions that have been added are:

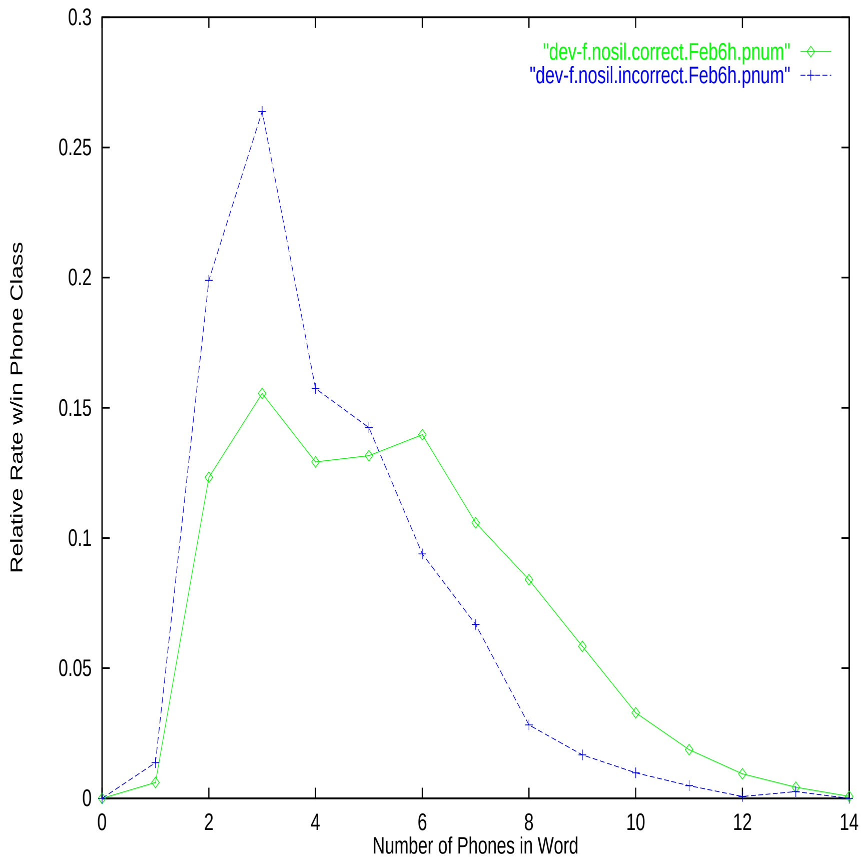


Figure 7.1: Relative rates within class of number of phones per word, broken out by phone classes {*correct, incorrect*}.

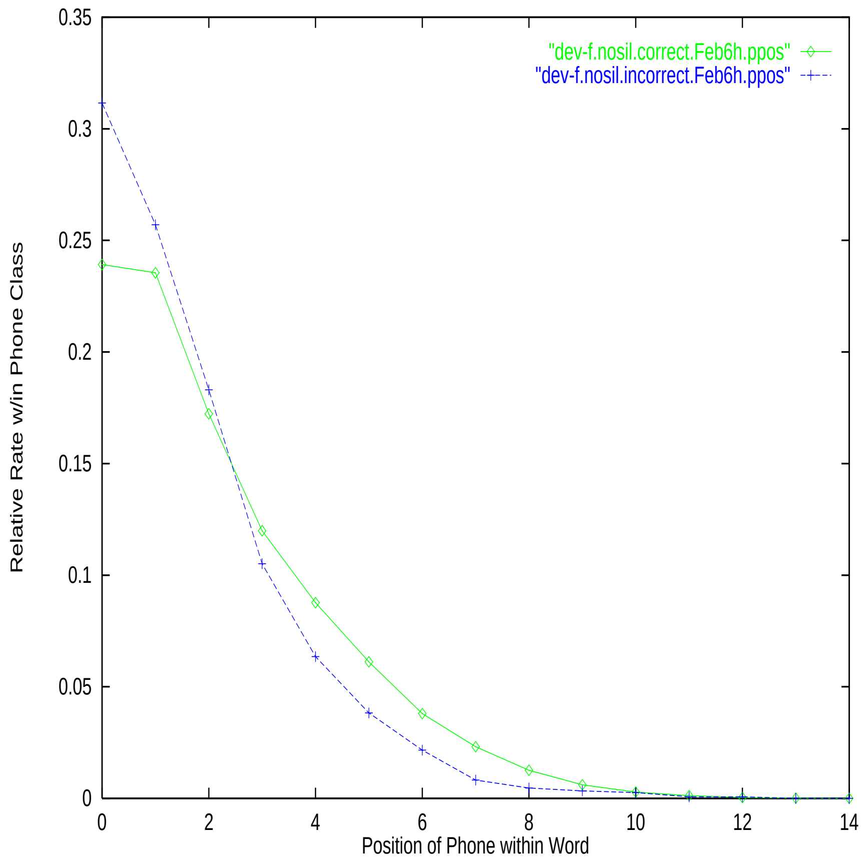


Figure 7.2: Relative rates within class of the position of phones in word, broken out by phone classes {*correct, incorrect*}.

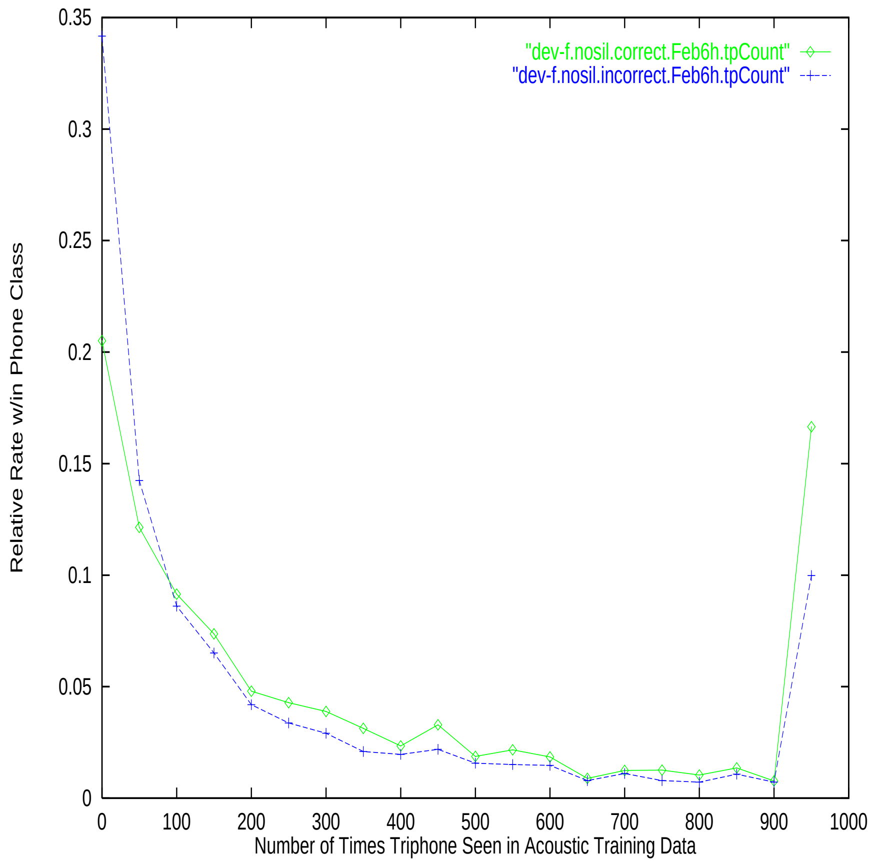


Figure 7.3: Relative rates within class of the number of times the triphone appeared in the acoustic training data, broken out by phone classes {*correct*, *incorrect*}.

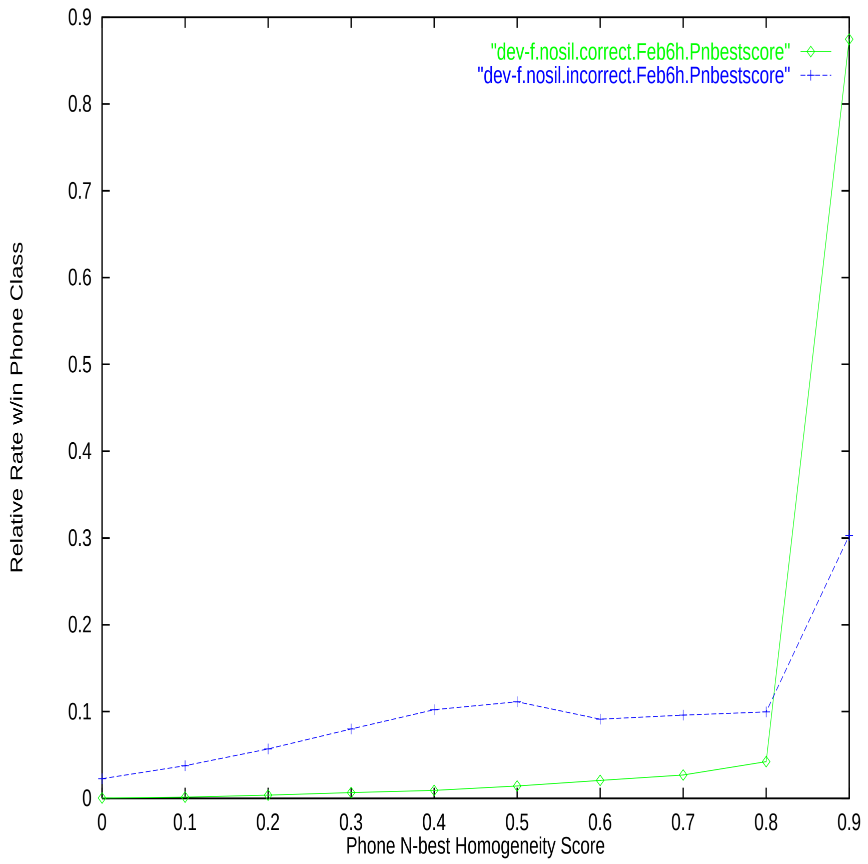


Figure 7.4: Relative rates within class of the N-best phone homogeneity score, broken out by phone classes {*correct, incorrect*}.

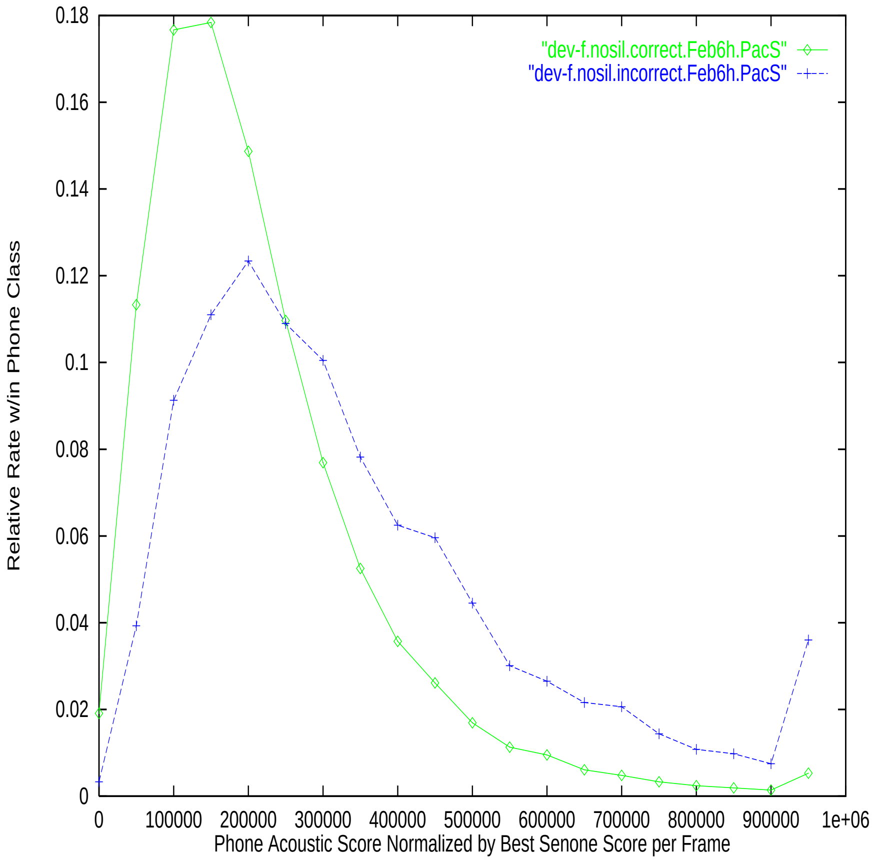


Figure 7.5: Relative rates within class of the phone acoustic score, normalized by best possible acoustic score, then broken out by phone classes $\{$*correct, incorrect*$\}$.

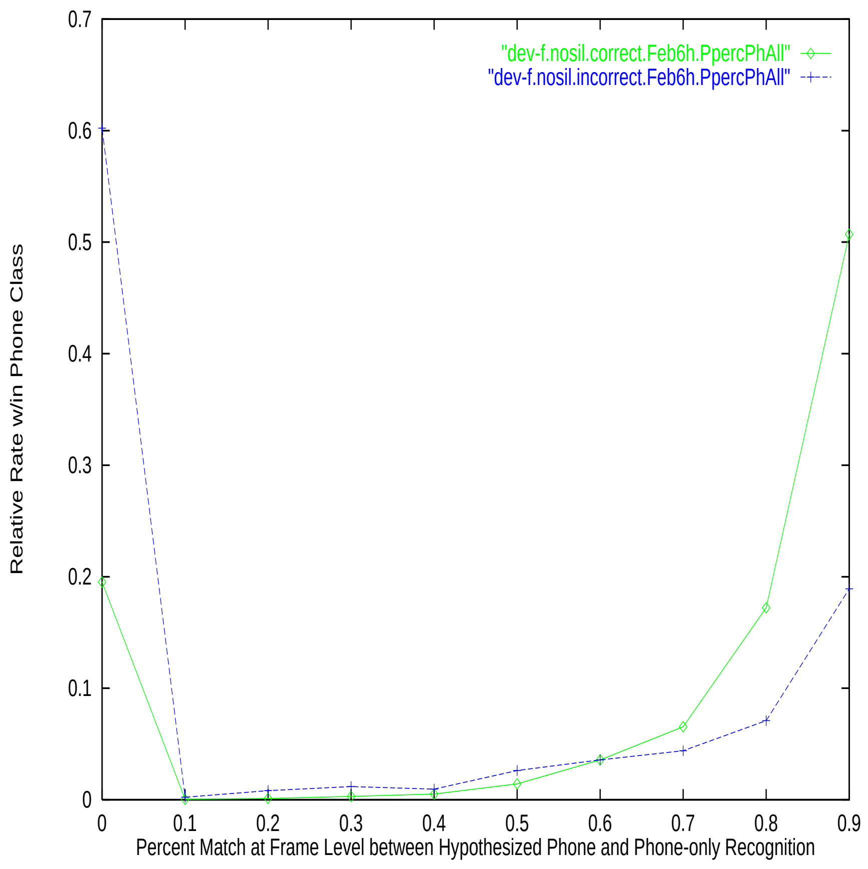


Figure 7.6: BASELINE: Relative rates within class of the frame match rate between HYPothesized phones and the phone-only decoding, broken out by phone classes {*correct, incorrect*}. This is the basis of the baseline system for the phone confidence annotation experiments.

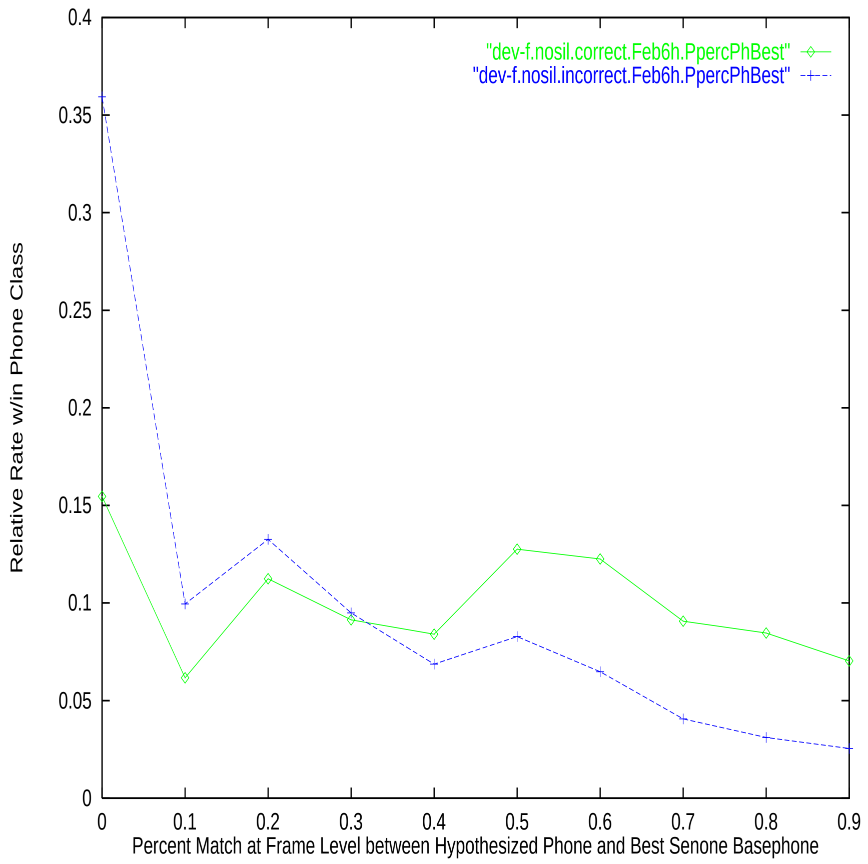


Figure 7.7: Relative rates within class of the frame match rate between HYPothesized phones and the best-senone basephone, broken out by phone classes {*correct, incorrect*}.

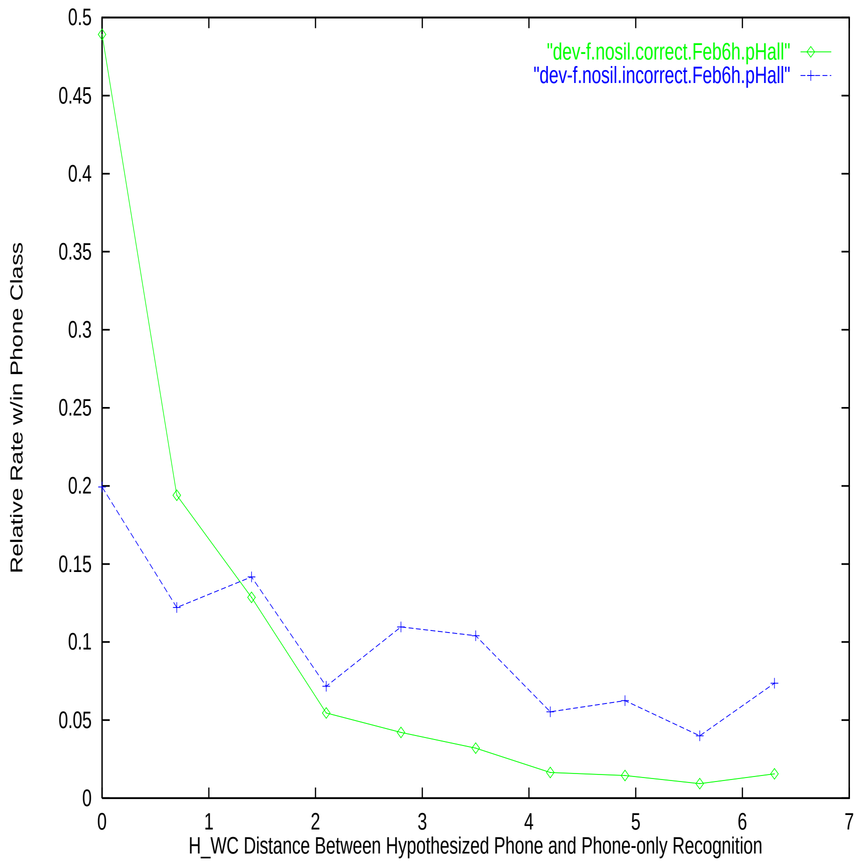


Figure 7.8: Relative rates within class of the H_{WC} distance between HYPothesized phones and the phone-only decoding, broken out by phone classes $\{$*correct, incorrect*$\}$.

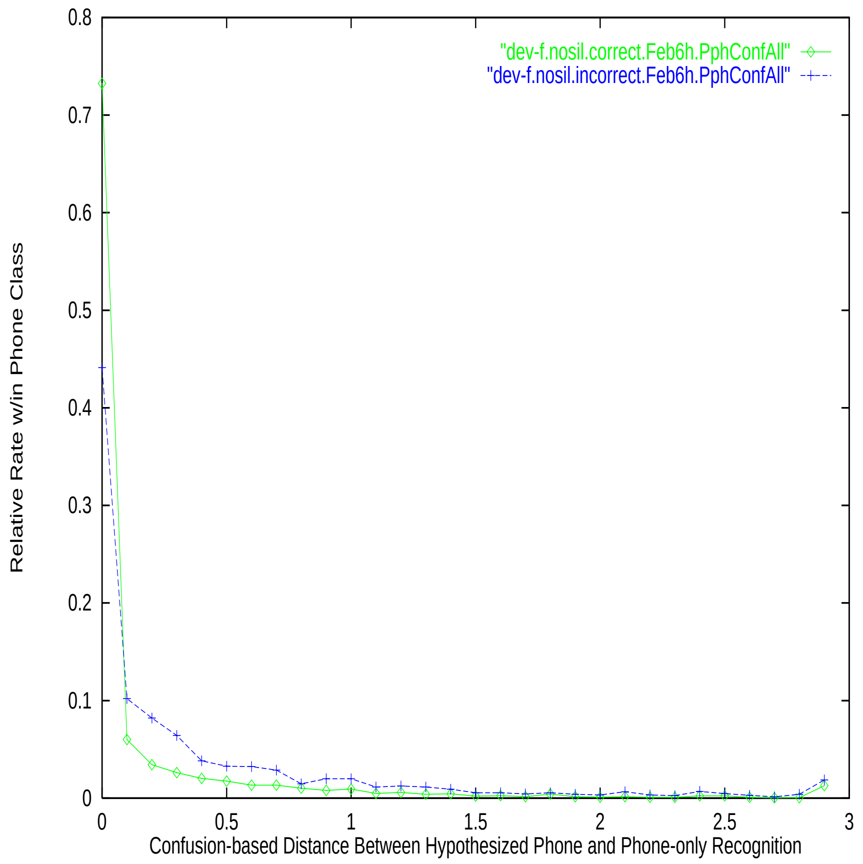


Figure 7.9: Relative rates within class of the confusion-based phone distance metric between HYPothesized phones and the phone-only decoding broken out by phone classes $\{$*correct, incorrect*$\}$.

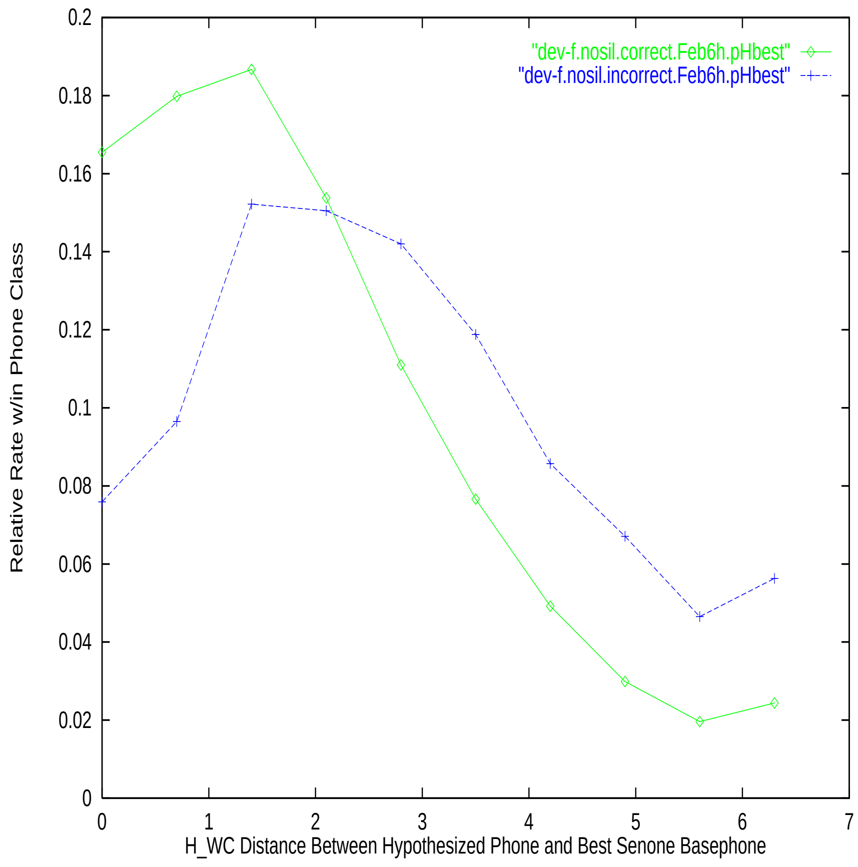


Figure 7.10: Relative rates within class of the H_{WC} distance between HYPothesized phones and the best-senone basephone at each frame, broken out by phone classes $\{$*correct, incorrect*$\}$.

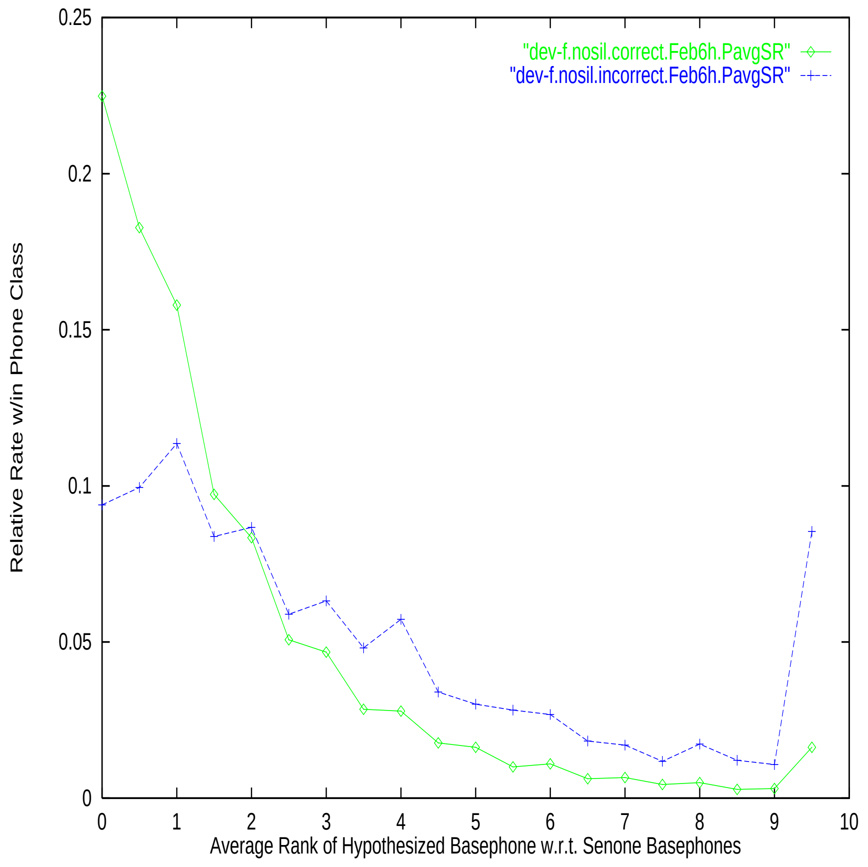


Figure 7.11: Relative rates within class of the frame-averaged rank of phone with respect to senone computations, broken out by phone classes $\{correct, incorrect\}$.

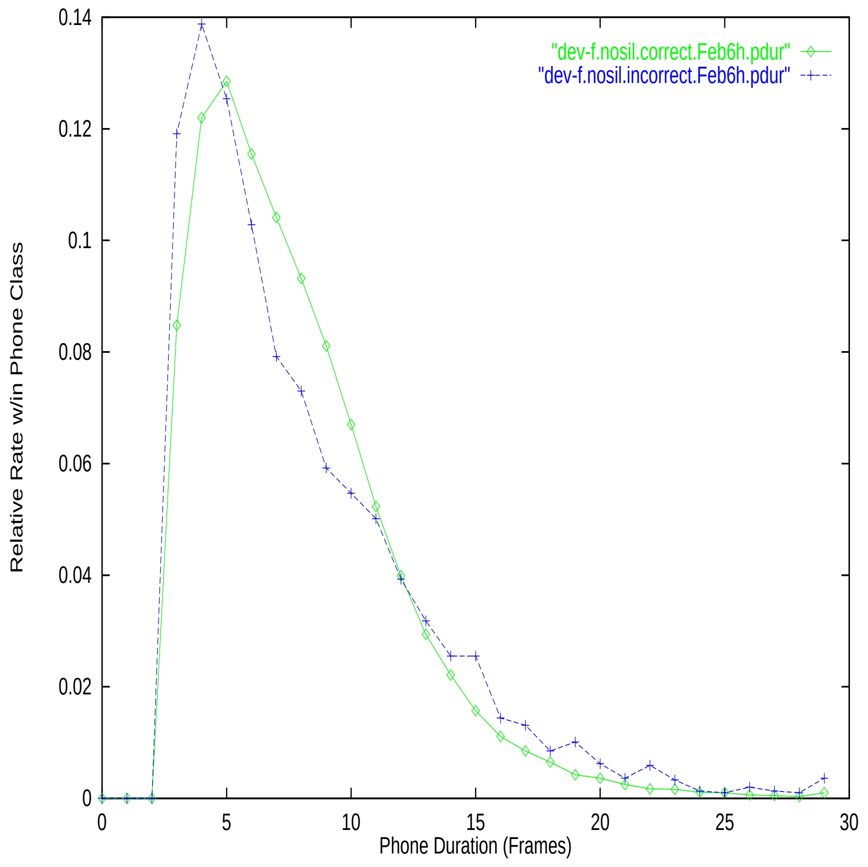


Figure 7.12: Relative rates within class of the duration in frames of the phone,broken out by phone classes {*correct*, *incorrect*}.

- *avgNbestHP*: A measure of how phonologically similar, under the H_{WC} measure, the phones in the N-best list are in the region of the frame segmentation of the phone in question. The time slice corresponding to the HYPothesized phone's segmentation is taken from each of the paths in the N-best list. Each of the resulting time segments is compared in turn under the H_{WC} measure with the HYPothesized phone. The average value is reported as the value of the predictor. See Figure 7.13. (None).

- *avgNbestConfP*: A measure of how typically confusable the phones in the N-best list are in the region of the frame segmentation of the phone in question. The computation proceeds as with $avgNbestHP$, except that the comparison measure used is the confusion-based phone distance metric, as in $ConfAllPhone$ and $ConfBestPhone$. See Figure 7.14. (None).

Figures 7.1 and 7.2 show that, as in the previous chapter's experiments, phones that appear in shorter words tend to be incorrect more often than phones that appear in longer words. These figures also show that the first phone of a word is more likely to be wrong than phones in other positions.

Figure 7.3 shows that mostly it doesn't matter how many times a triphone was seen in the acoustic training data, the phone was never seen at all. Unseen triphones have a higher likelihood of being wrongly hypothesized than do those that were actually trained on data. The same is not true for basephones, however. There is no correlation between likelihood of correctness and number of times a basephone was seen in training. (No figure is shown for this variable.)

Figure 7.4 indicates that NbestPhone, like its word-level counterpart Nbest, is an excellent predictor of correctness of phones. In fact, in comparing this figure with the previous chapter's Figure 6.3, we see that the effect of separation is even more pronounced at the phone level than at the word level, especially for low $P(C)$ values.

Another predictor variable whose separation power increases dramatically at the phone level is acNormSphone, whose word-level analog is acNormS. Figure 7.5 shows the very clear pattern that phones that are hypothesized correctly have acoustic scores much closer to the optimal unconstrained scores than do incorrectly hypothesized phones.

Comparing basephone matches between the hypothesized phone and the phone-only decoding also works in an improved fashion at the phone level,

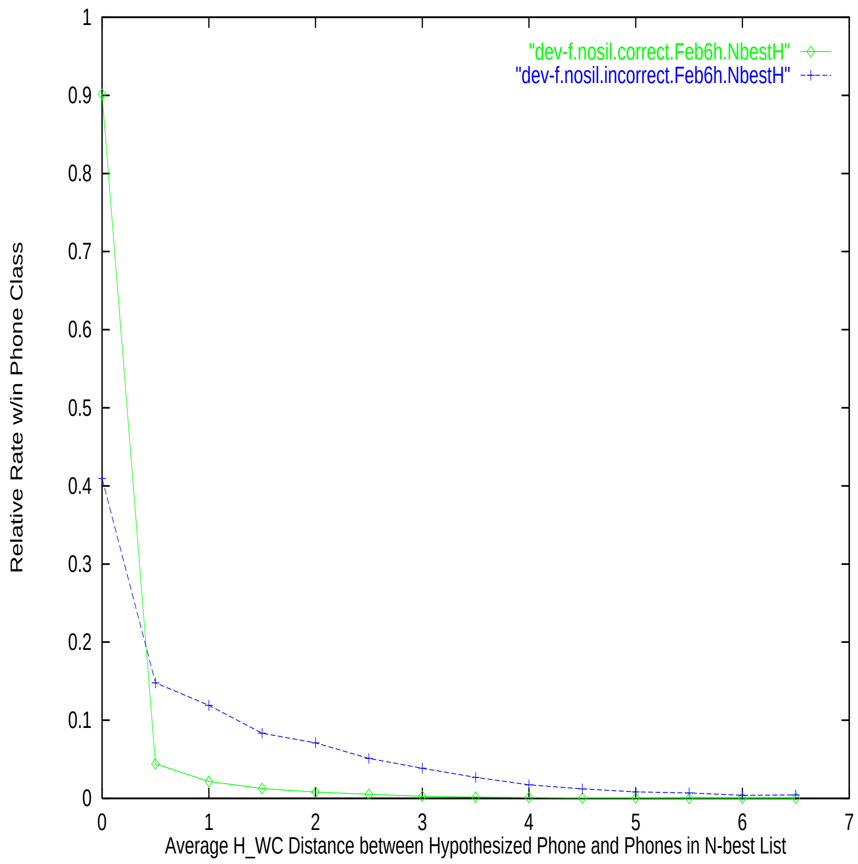


Figure 7.13: Relative rates w/in class of the N-best phone similarity measure $avgNbestHP$ broken out by phone classes $\{correct,\ incorrect\}$.

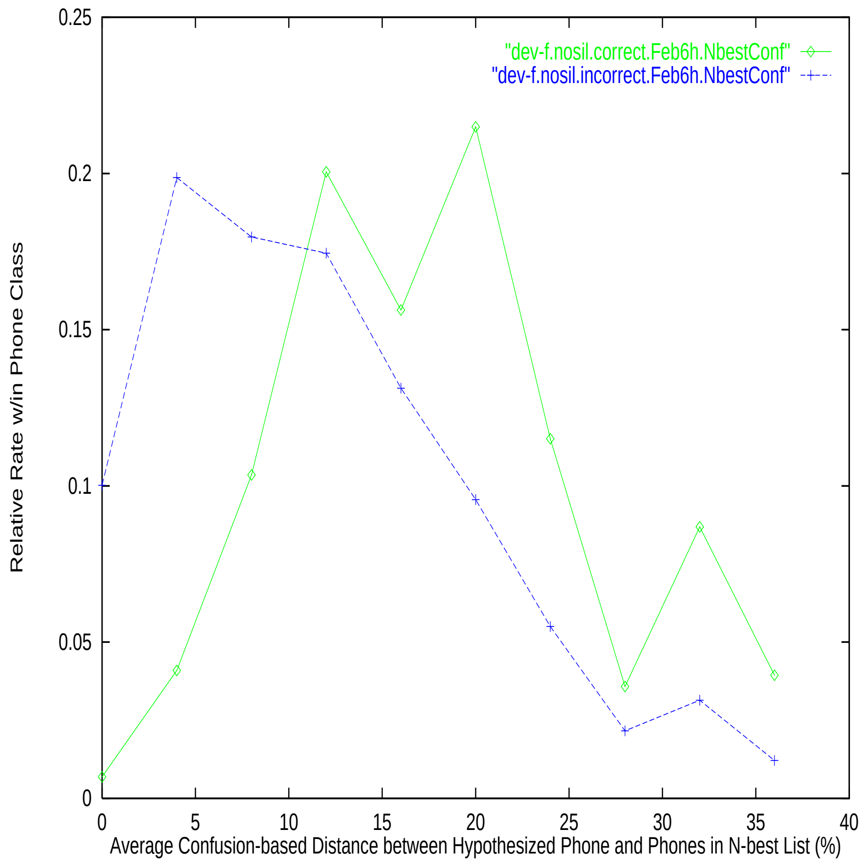


Figure 7.14: Relative rates w/in class of the N-best phone confusability measure $avgNbestConfP$ broken out by phone classes $\{$*correct*, *incorrect*$\}$.

as indicated by the nicely separable curves in Figure 7.7. The general configuration of these curves comes as close to the ideal any seen so far; the peak at the left end of the scale for incorrect phones opposes the peak at the right end for correct phones very nicely.

Unlike in the previous word-level experiment, counting matches at the phone level between the hypothesized phone and the best-senone basephone actual does create a class separation, as in Figure 7.7. Recall that in the word-level experiments the analagous measure produced no useful separation between classes.

Figures 7.8 and 7.9 show the results of using the H_{WC} distance measure and the confusion-based measure with the phone-only recognition as a reference point. As in the previous example, the H_{WC} gives some separation and the confusion-based metric not much at all.

A sizeable (and surprising in contrast with the word-level) result was achieved when the H_{WC} metric was used in conjunction with the best-senone reference point, as in Figure 7.10. The use of the confusion-based metric, however, continued to fail to give any separation in conjunction with the best-senone basephone.

The confusion-based metric, however, turns out to be a very useful distance measure when used as in the predictor avgNBestConfP, which measures how typically confusable the set of phones found at a particular time slice in the N-best list are in comparison with the hypothesized phone. Figure 7.14 indicates a very nice separation between the two classes using this predictor. Some separation is also created with the related predictor NbestHP, which uses the phones in the N-best list in the same way but relies instead on the H_{WC} distance measure. (See Figure 7.13).

Average senone rank again turned out a good separation for low values of the rank score, as shown in Figure 7.11.

Duration and acoustic score of the phone (neither normalized in any way), however, failed to produce anything useful.

7.3 Performance of the Predictors (ν)

In this phone-level experiment we define a baseline analagous to the BASE1 system of the previous chapter. This baseline is simply the predictive power of the $BASE : phAll$ predictor, which captures the rate at which the HYPothesized phones match those found in the phone-only decoding. (No sec-

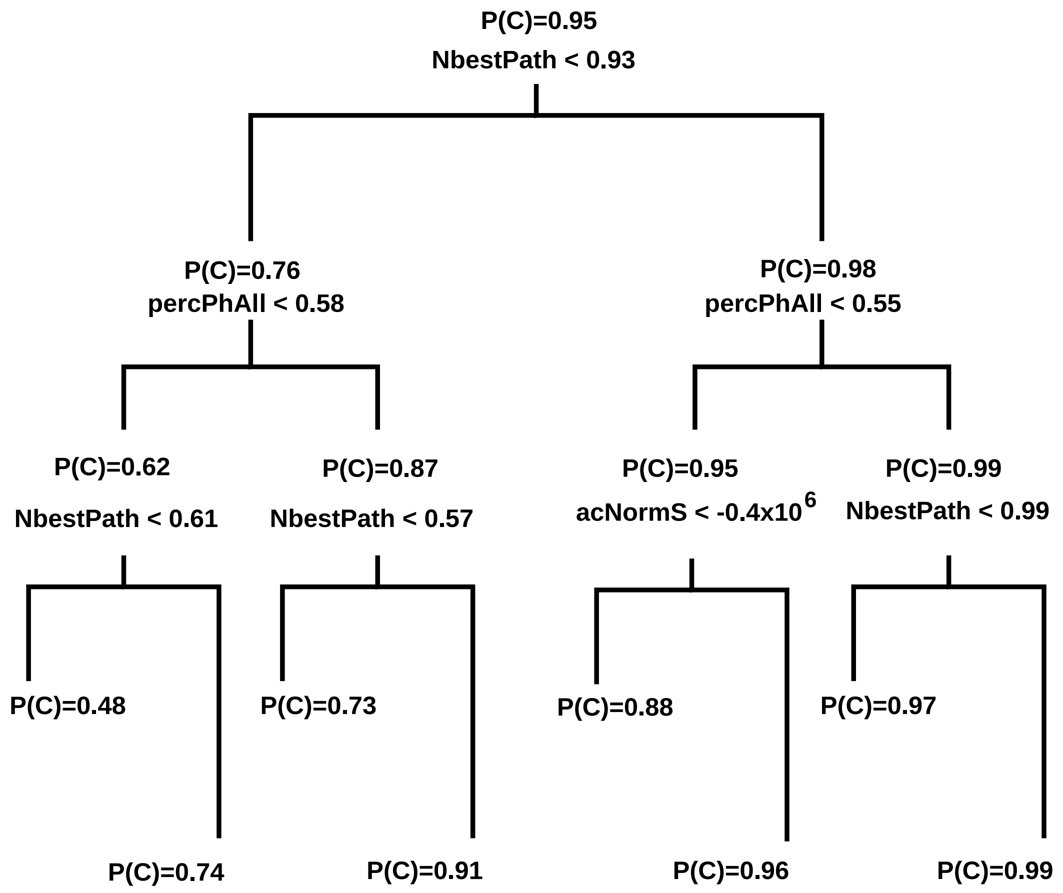


Figure 7.15: Top branches of the best female phone-level predictor of error labels {*correct, incorrect*}. NAB data.

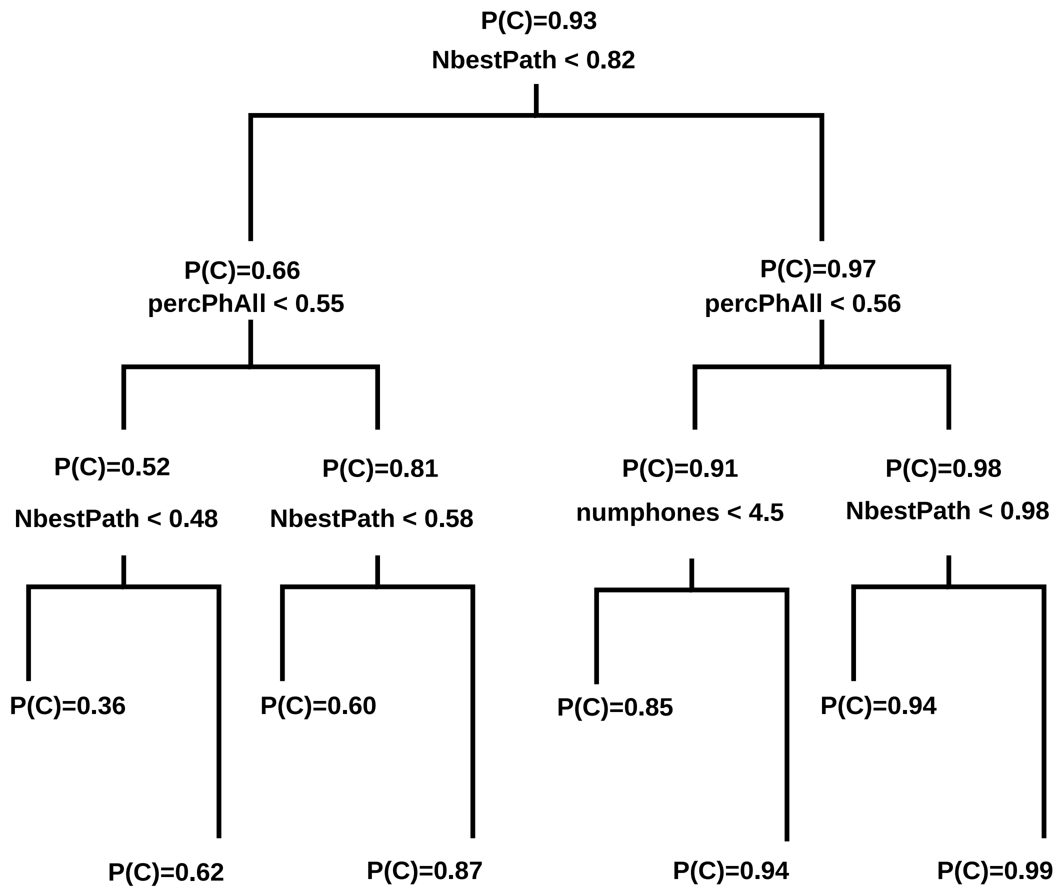


Figure 7.16: Top branches of the best male phone-level predictor of error labels {*correct, incorrect*}. NAB data.

ond baseline based on combined word score is possible, as language model information is not used in this experiment.)

The base rate of *incorrect* phones in the female experiment is 6%. The base rate in the male experiment is 7%. These experience rates are between 2 and 3 times *less* frequent than the word rates from the previous experiments.

Predictor Variables	% X-entropy (female)	% X-entropy (male)
BASE:percPhAll	11.2%	12.2%
percPhAll+Hall+ConfAll	11.8%	13%
percPhBest+Hbest+ConfBest	4.9%	5%
senRank+acNormS	6.6%	7.5%
NbestPath+NbestH+NbestConf	25%	22.6%
tpCount+phRel	2.6%	2.1%
numphones+position	4%	2.9%
Best System	29.1%	29.3%

Table 7.1: Improvement in 2-class phone confidence annotation as a function of the predictor variables used. For both female and male acoustic models.

Table 7.1 summarizes the relative power of subsets of the predictor variables used. Graphic summaries of the figures presented in Table 7.1 are given in Figures 7.17 and 7.18.

By far the best predictors of phone correctness are those based on the N-best analyses. The combination of NbestPath, NbestH, and NbestConf yields a total of 25% (22.6%) reduction in cross-entropy for the female (male) models. Their counterparts in the word-level experiment yielded only 12%. This is remarkable, especially in light of the fact that the *incorrect* phones appear 2 to 3 times less frequently than did *incorrect* words!

Comparison of the hypothesized phones with those found in the phone-only decoding also yielded considerable cross-entropy reductions. The baseline system, which does only a simple frame-level match, supplies an 11.2% (12.2%) reduction in cross-entropy when working alone. The additional contributions of the slightly more sophisticated H_{WC} and confusion-based metrics increase this to 11.8% (13%), which is consistent with what we saw in the previous word-level experiment.

The relative contribution of the senone and acoustic score contributions, senRank+acNormS, are twice the strength of their word-level counterparts.

The numphones+position combination also significantly approves with respect to the word-level version.

Training counts, however, produce a very similar, and thus small, effect.

Figure 7.15 shows the first few interactions between predictor variables in the best system. The first variable, NbestPath, identifies phones for which the evidence in the N-best indicates some possible confusion on the part of the recognizer. Possible confusion indicates a lower likelihood of correctness for the phone.

The next adjustment made to this is based on a comparison of the hypothesized phone with the phone-only decoding. The results of the comparison can either increase (if the match is good) or decrease (if the match is not good) the likelihood that a phone is *correct*. If the phone was originally suspect due to possible confusions found in the N-best list, then a good match with the phone-only decoding can restore the probability of the phone being *correct* to a high value. A mismatch on an already suspect phone, however, decreases the likelihood of correctness substantially.

Next, acNormS is used to distinguish *correct* and *incorrect* phones from each other when the Nbest list is not very confused in the region of the phone, but the phone-only recognizer does not agree with the hypothesized phone. In these cases those phones whose acoustic scores don't differ much from the best possible acoustic score retain their high likelihood of being *correct*, while those whose acoustic scores diverge from the optimal become suspect, and receive a lower $P(C)$.

This pattern of main questions is the same for both the female and male experiments. (See Figure 7.16.)

Overall the best system produces a 29.1% (29.3%) reduction in cross-entropy. This is a dramatic improvement over the word-level results, especially in light of the greatly reduced base rate of class *incorrect*.

The probability-of-correct ($P(C)$) curves produced by the baseline annotator for male speakers are shown in Figure 7.19. Thresholding these $P(C)$ curves gives us False Alarm/Missed Error rates, as shown in Figure 7.20. The curves produced by the baseline annotator on female speakers are extremely similar, and thus are not shown. (The same is true for their corresponding FA/ME curves.)

The probability-of-correct ($P(C)$) curves produced by the best annotator for both male and female speakers are shown in Figure 7.21 and 7.22. The False Alarm/Missed Error rates that correspond to these $P(C)$ curves are shown in Figures 7.23 and 7.24. Note that if a relatively low threshold on

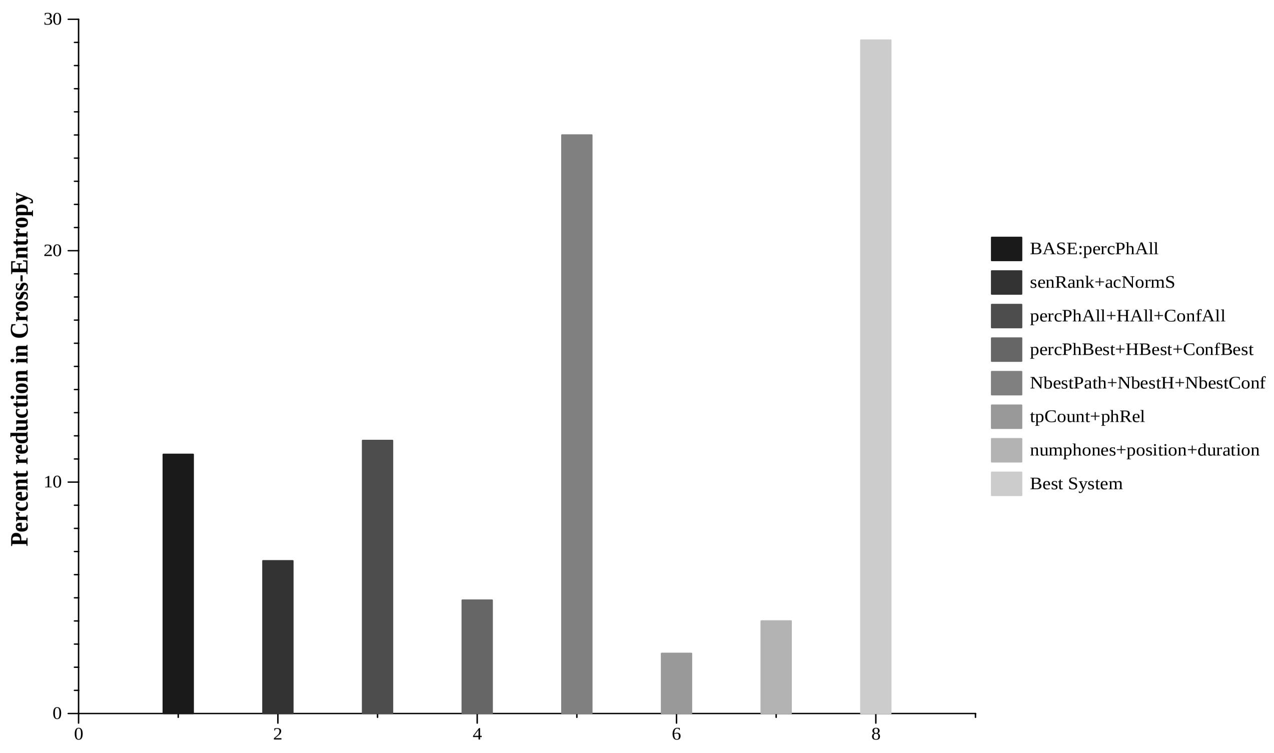


Figure 7.17: Incremental confidence annotation power from various subsets of the phone predictors. Female acoustic models on female speakers. Performance measured in percent reduction in cross-entropy of labeling classes {*correct, incorrect*}.

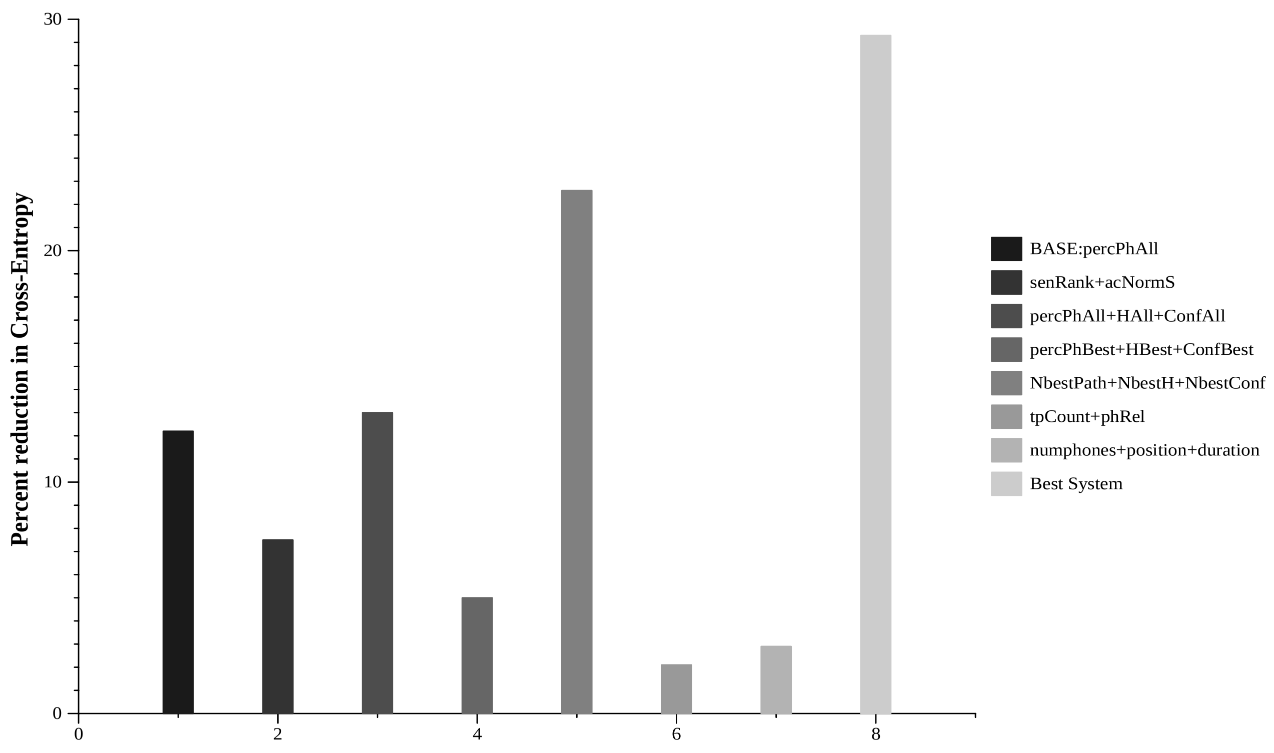


Figure 7.18: Incremental confidence annotation power from various subsets of the phone predictors. Male acoustic models on male speakers. Performance measured in percent reduction in cross-entropy of labeling classes {*correct, incorrect*}.

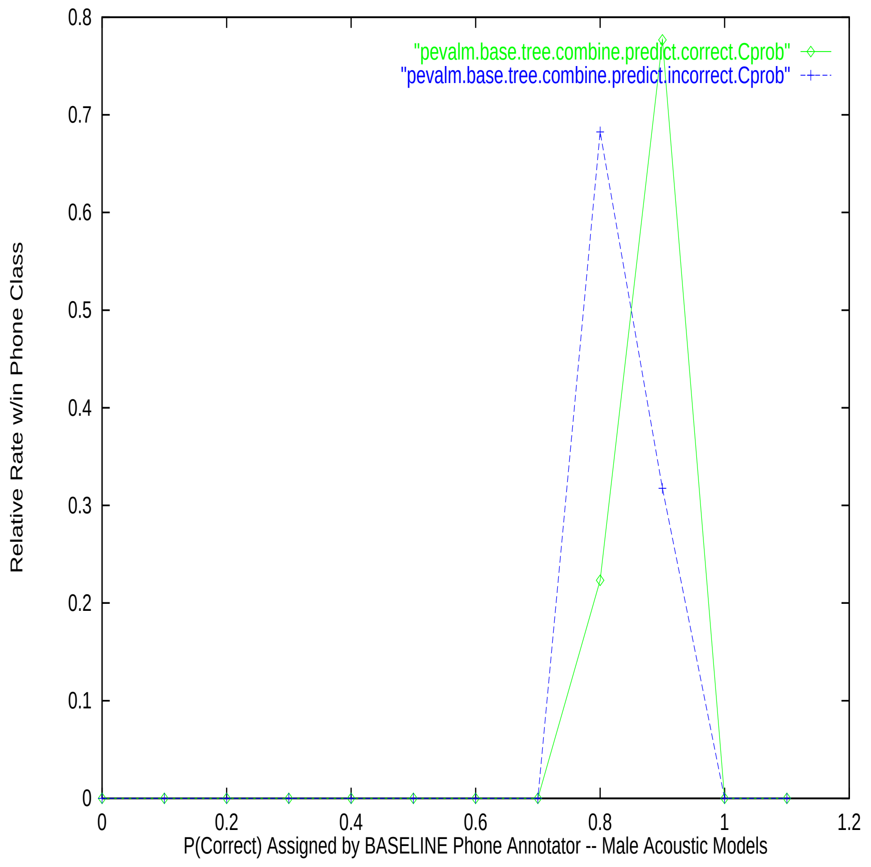


Figure 7.19: The BASELINE phone-level predictor of phone error classes $\{$*correct, incorrect*$\}$. Male acoustic models on male speakers.

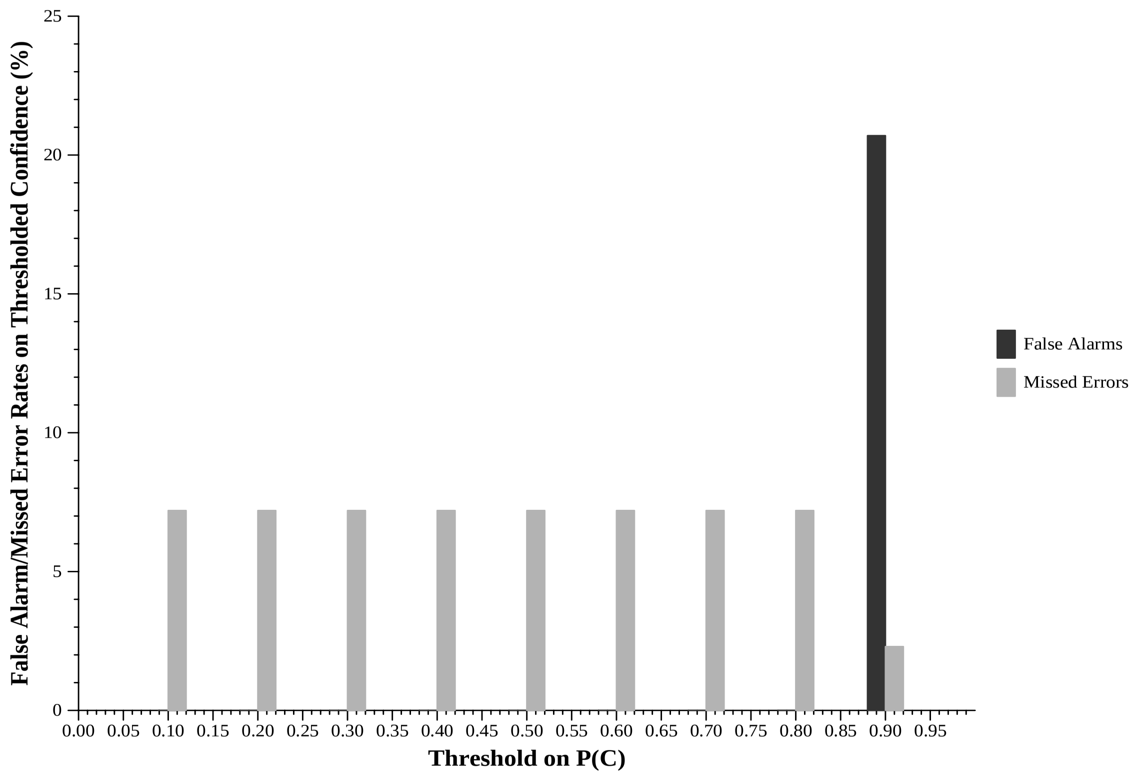


Figure 7.20: False Alarm/Missed Error rates at various thresholds of $P(C)$ for the BASELINE system on male acoustic models.

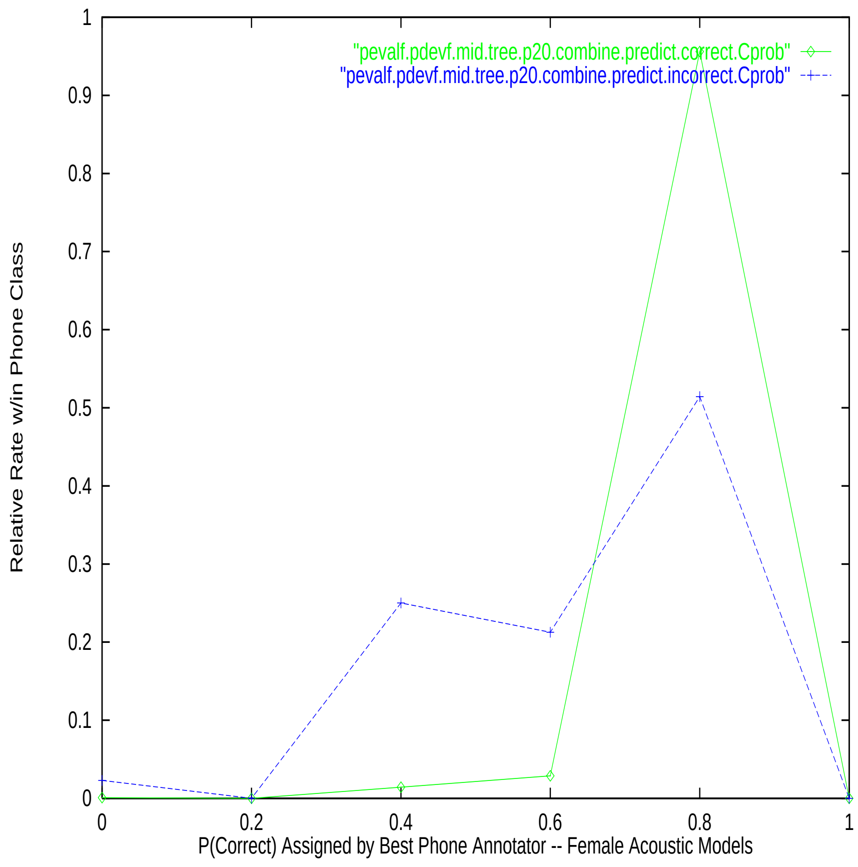


Figure 7.21: The best phone-level predictor of phone error classes {*correct*, *incorrect*}. Female acoustic models on female speakers.

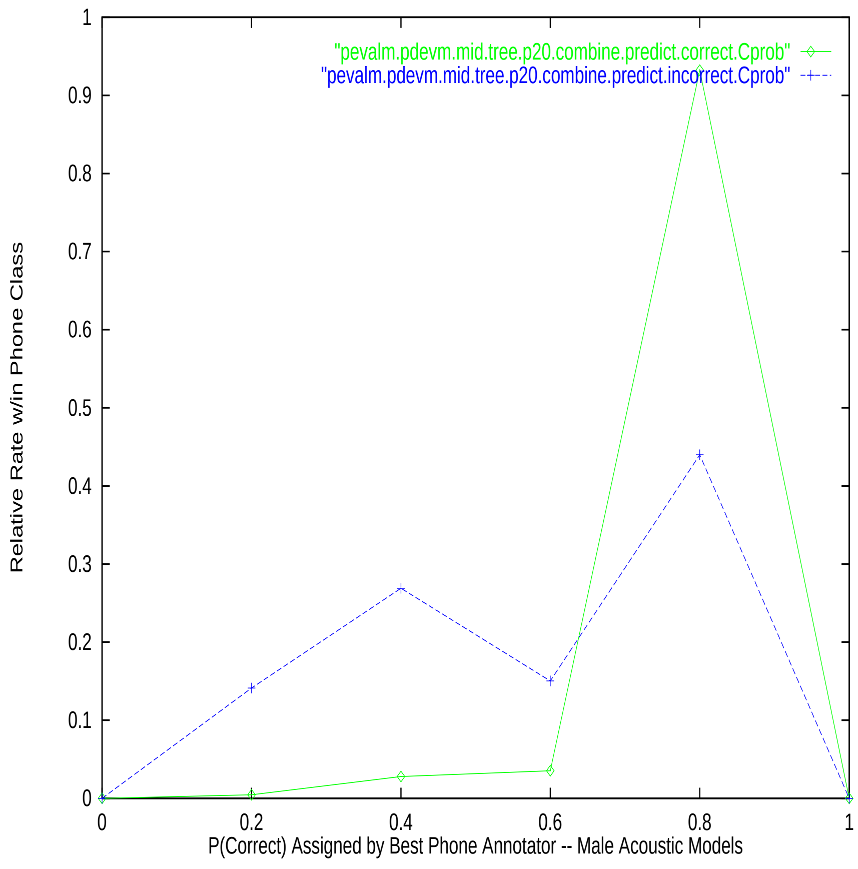


Figure 7.22: The best phone-level predictor of error labels {*correct*, *incorrect*}. Male acoustic models on male speakers.

$P(C)$ of between 0.5 and 0.6 is used, that approximately 30% of the *incorrect* phones can be identified while suffering a very low corresponding false alarm rate.

7.4 Summary

In this chapter we revisited the NAB confidence annotation experiments. Instead of doing word-level confidence annotation, however, we moved down a level of abstraction to the phones. The results is a powerful acoustic confidence annotator that can produce a cross-entropy reduction of more than 29% in the identification of incorrectly hypothesized phones. In comparison with the word-level results of 21%, this is remarkable, as the base rate of *incorrect* phones is 2 to 3 times less frequent for phones than for words.

Most of the predictor variables used in this experiment were phone-level analogs of acoustically-motivated predictor variables from the previous chapter's experiments. Two new predictors were added, however. They are phone-level refinements to the most powerful predictor from the previous chapter.

In almost every case the phone-level version of the predictor variable was able to separate classes *correct* and *incorrect* more readily.

The strongest predictor variable, NbestPath, is the first found in the best annotator's decision tree. It identifies phones for which the evidence in the N-best indicates some possible confusion on the part of the recognizer. Possible confusion indicates a lower likelihood of correctness for the phone.

The next adjustment made to this is based on a comparison of the hypothesized phone with the phone-only decoding. The results of the comparison can either increase (if the match is good) or decrease (if the match is not good) the likelihood that a phone is correct. If the phone was originally suspect due to possible confusions found in the N-best list, then a good match with the phone-only decoding can restore the probability of the phone being correct to a high value. A mismatch on an already suspect phone, however, decreases the likelihood of correctness substantially.

Next, acNormS is used to distinguish correct and incorrect phones from each other when the Nbest list is not very confused in the region of the phone, but the phone-only recognizer does not agree with the hypothesized phone. In these cases those phones whose acoustic scores don't differ much from the best possible acoustic score retain their high likelihood of being *correct*, while those whose acoustic scores diverge from the optimal become suspect,

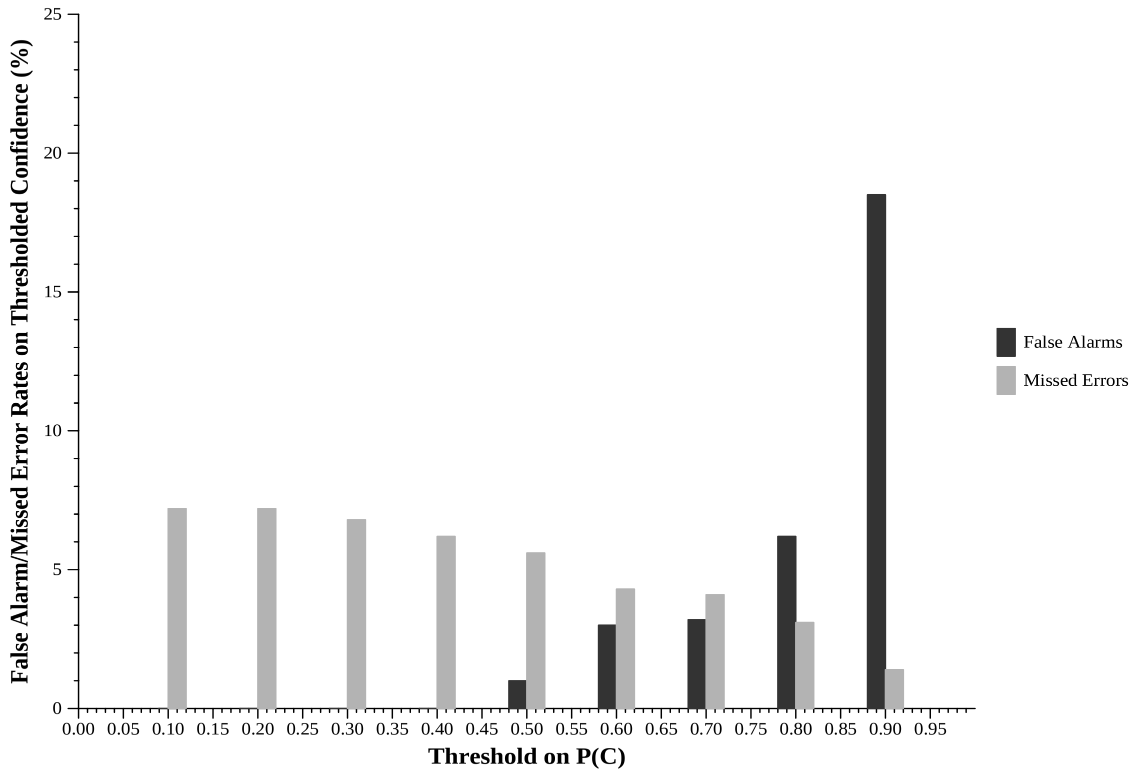


Figure 7.23: False Alarm/Missed Error rates at various thresholds of $P(C)$ for the best system on male acoustic models. Phone confidence annotation, NAB data.

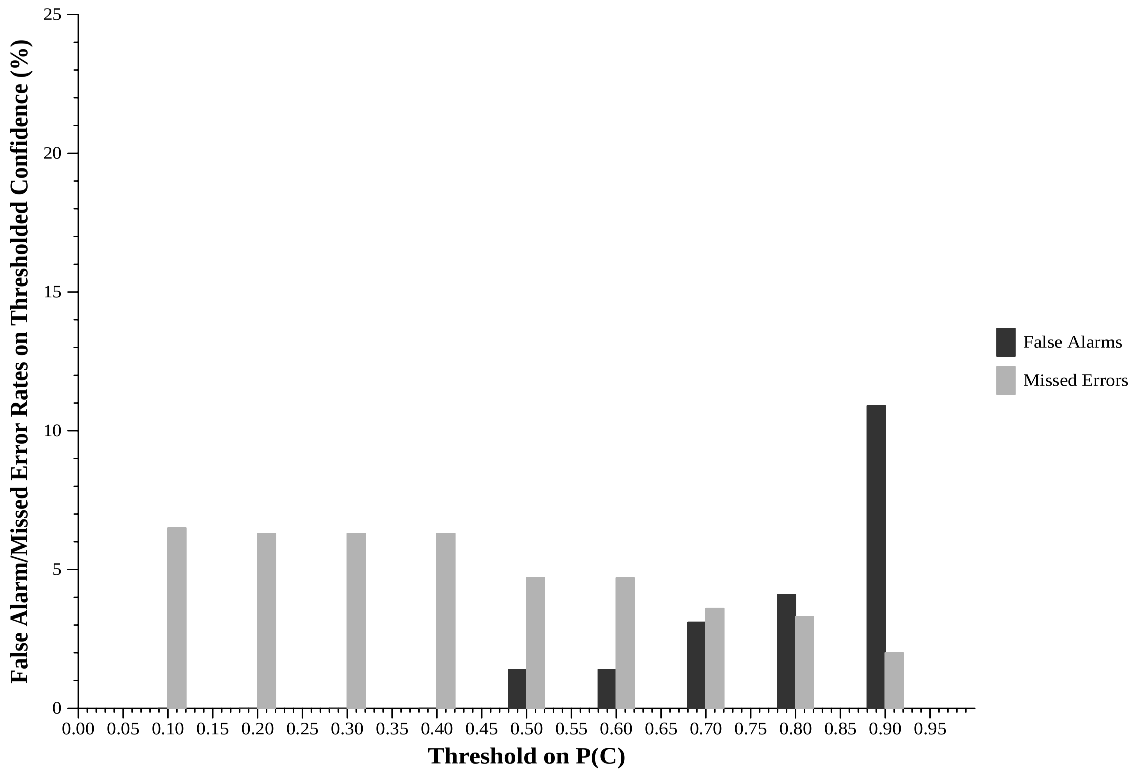


Figure 7.24: False Alarm/Missed Error rates at various thresholds of $P(C)$ for the best system on female acoustic models. Phone confidence annotation, NAB data.

and receive a lower $P(C)$.

Other predictors that contribute in lower branches of the tree include the number of phones in the word hypothesized, the distance between the hypothesized phone and the phone-only decoding measured with the H_{WC} phonological metric, the number of times the triphone was seen in the training data, and the position of the phone in the hypothesized word. This pattern of questions is the same for both the female and male experiments.

In the next chapter we apply what we learned about the strength of predicting errors at the phone level by including the phone-level predictors in a word confidence annotation experiment.

Chapter 8

Word Confidence Annotation for Conversational Telephone Speech

In this chapter we revisit the issue of word-level confidence annotation. In these experiments we will provide annotations to telephone conversations held between two speakers.[1] In addition to switching recognition domains we will also switch speech recognition systems. The Janus system, which has been primarily developed for conversational rather than read speech tasks, is used instead of Sphinx-II. The database used is called "Switchboard".

Instead of predicting three classes as in Chapter 6, we will predict only two:

1. *correct*: the speech recognizer guessed the correct word according to a word-level alignment of the hypothesized word sequence against the correct transcript, or

2. *incorrect*: the alignment of hypothesized output against the correct transcript indicated an error for the word.

The problem of OOV words in the Switchboard task is of relatively less importance than it is in the large vocabulary dictation task, as the typical error rates on Switchboard are close to 40% WER, as compared to 15%

[1]As described in Chapter 3, the speakers do not know each other and they have been assigned a topic for discussion.

on the NAB task. Thus the refinement into two categories of error is less urgent, and the search for better predictors of error in general and insight into system design problems overall is more critical.

Recall that we found in the word confidence annotation experiments on read speech that by predicting two types of error separately (for a total of three classes) we were able to marginally increase our overall error prediction performance. Since we are not modeling errors due to OOVs in this experiment we potentially lose that source of predictive power.

The set of predictor variables discussed in this chapter is similar to those found in Chapter 6. The predictor variables are constructed in reference to the same important sources of information, including:

- Word and phone segmentations and acoustic scores from the best-scoring HYPothesis produced by the recognizer,
- Language model score information from the same best-scoring HYPothesis,
- Characteristics about the words HYPothesized, including their pronunciations and how frequently they appear in acoustic training materials,
- The results of a completely unconstrained frame-by-frame decoding in which the best possible acoustic score and basephone are determined for each frame of the utterance,
- Two distance metrics between basephones, including a simple match count at the frame level, and a phonologically-based similarity measure (H_{WC}),
- The contents of an N-best list that contains complete word and phone segmentation and score information for each of 150 elements.

No phone-only decoder was available for use with Janus, so we instead rely on the best-scoring basephone at each frame as our external phone reference.

No confusion-based distance metric was calculated on the Switchboard data. We rely instead on the remaining two phone distance metrics: percent match and H_{WC}.

No language model score source tags were available from the Janus system.

As the Janus acoustic models are essentially continuous Gaussian models, and because we calculate only the best score at each frame and not a long vector of these values, no "senone ranking" predictors are available for this experiment.

Because, as described in Chapter 2, we can easily have access to Janus lattices and not just N-best lists, an additional type of predictor variable is available to us for these Switchboard experiments. Normally an empirically optimized combination of language weight (LW) and insertion penalty (IP) are used to search a lattice in order to produce a hypothesis. If, however, a wide range of language weights and insertion penalties are tried, it is possible to count how often a particular word appears and disappears from the hypothesis with the changing LW/IP pairs. This number, which varies between 0 and 1, is called "LM Jitter" below, and is a very useful predictor of error.

The descriptions of those predictors that depend on the details of how acoustic scores are calculated have been modified to suit the Janus acoustic modeling scheme. In all cases figures describing the relative effectiveness in class separation between *correct* and *incorrect* are shown.

As in the previous experiment, both domain-independent and application-specific evaluation metrics are discussed. A detailed comparison between decision trees and three other learning mechanisms is also included.

The data used in these experiments is the Switchboard data described in Chapter 3. The curves shown for each predictor variable come from the development data, which serve as training data for the confidence annotator. All confidence annotation results are reported for the independent evaluation test data.

8.1 Predictor Variables Used (ν)

In describing which predictor variables were used and what their relative strengths are, we will refer to each with the following reference tags:

- *BASE:percBestCor*: Somewhat analagous to the previous baseline systems based on percentage match among the phones in the hypothesized word and the phone-only decoding. Here, however, a smoothed match[2]

[2]The smoothing algorithm is described later in this section.

is computed with the best-scoring basephone at each frame, as phone-only decodings are not available. See Figure 8.1.

- *phAvgpercBestCor*: The percBestCor figure is calculated for each phone in the word, then averaged within word over the number of phones in the word. See Figure 8.2.

- *phMinpercBestCor*: The percBestCor figure is calculated for each phone in the word. The minimum value among the phones in the word is reported. See Figure 8.3.

- *LMscore*: The word language model score, including language weight and insertion penalty factors. See Figure 8.8.

- *Nbest*: The N-best homogeneity score computed at the word level with a frame tolerance of three. See Figure 8.5.

- *phNBestAvg*: The phone-level N-best homogeneity score computed for each phone in the word and then averaged within the word over the number of phones.

- *phNBestMax*: The phone-level N-best homogeneity score is computed for each phone in the word. The maximum value found for a phone in the word is reported.

- *phNBestDel*: The difference between phNbestAvg and phNbestMax. See Figure 8.6.

- *avgWF*: The frame-averaged number of words found in the N-best list at the location of the hypothesized word. See Figure 8.7.

- *maxWF*: The maximum number of words found in the N-best list within the segmentation of the hypothesized word.

- *delWF*: The difference between avgWF and maxWF for the word in question.

- *numphones:* The number of basephones found in the word's dictionary pronunciation. See Figure 8.4.

- *numshort*: The number of minimal-length (3 frame) phones found in the word. See Figure 8.15.

- *avgHPBest*: The average value of the H_{WC} metric of the word's phone compared with the best-scoring basephone at each frame in the word's segmentation.

- *phAvgHPBest*: The avgHPBest calculation, only performed first at the phone level and then averaged within the word over the number of phones in the pronunciation. See Figure 8.12.

- *phMaxHPBest*: The avgHPBest calculation is performed for each phone in the word. The maximum phone-level value found within the word is reported. See Figure 8.13.

- *acNormS*: The ratio between the best possible acoustic score within the segmentation of the word and the actual acoustic score of the word. Expressed in terms of percent difference in log probability with respect to the best score. See Figure 8.9.

- *AvgPhAcNormS*: acNormS calculated for each phone in the word, then averaged over the number of phones in the word. See Figure 8.10.

- *DelNormS*: The difference between acNormS and AvgPhAcNormS. Captures whether there is a big local difference between the phone-level and word-level normalizations of acoustic scores within the word. See Figure 8.11.

- *phAvgHPBest*: The H_{WC} distance is calculated between each phone in the word and the best-scoring basephone at each frame. This is then averaged within the word over the number of phones. See Figure 8.12.

- *phMaxHPBest*: The H_{WC} distance is calculated between each phone in the word and the best-scoring basephone at each frame. The maximum phone value within the word is reported. See Figure 8.13.

- *TrainWC*: Count of number of times the word was seen in the acoustic training data.

- *TrainBC*: Count of number of times the basephone was seen in the acoustic training data, broken out into four word positions (single, begin, mid, end).

- *Duration*: The duration in frames of the word.

- *WordER*: The rate at which a particular word has been found to be *correct* in the development data.

- *phHWCNBestMax*: Taking a time slice in the N-best list that corresponds to the segmentation of the word, determine the maximum distance between an hypothesized phone and the N-best list phones under the H_{WC} metric. See Figure 8.14.

- *LMJitter*: Percentage of times the word remains present in the hypothesis under varying language weight and insertion penalties while decoding the utterance lattice. See Figure 8.16.

- *prevSil*: Whether or not the segment preceding the word in the hypothesis was either a silence or noise word, as opposed to a lexical item.

- *prevPC*: The confidence measure assigned to the previous segment.

As we have no Janus phone-only decodings, the external reference for the correct phone at any given frame comes only from the best-scoring basephone at each frame. Because this method of estimation is completely unconstrained between frames, the sequence of best basephones is often noisy when compared with the aligned reference sequence. To make the best basephone sequence a more reliable reference point we smooth it within a frame tolerance of two frames. For instance, if we are comparing a hypothesized phone at frame t with the best basephone sequence, we will give a match score of 1.0 if we find a match in the best basephone sequence at frame t. If we do not find a match at the same frame, but rather at either $t-1$ or $t+1$ in the best basephone sequence, we will give a match score of 0.9. If a match is not found within this one-frame window, but rather at either $t-2$ or $t+2$ in the best basephone sequence, then we will give a match score of 0.8.

The results of this smoothed match distance metric are presented in Figures 8.1, 8.2, and 8.3. These results are far better than their counterparts in the NAB experiments were. (Recall that in the NAB experiments no distance metric gave good separation results with the best basephone reference.) This is partly due to the smoothing employed, but mostly due to the superior acoustic models used in Janus.[3]

[3]The Janus acoustic models have a great many more free parameters than the Sphinx-II models. They also run approximately 100 times slower on the same machine.

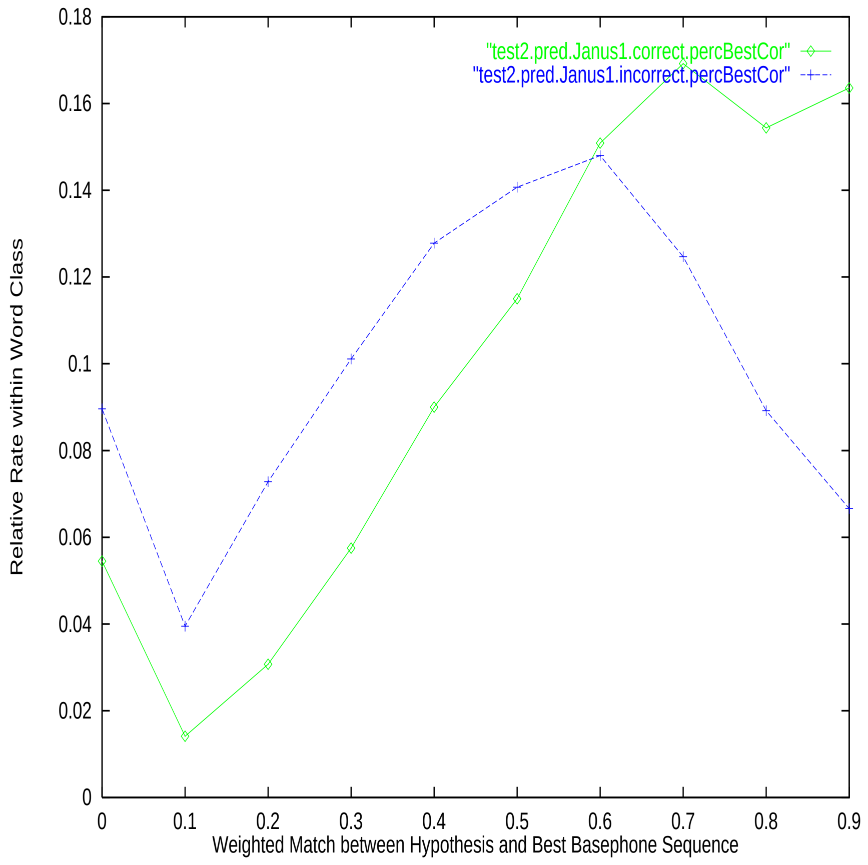


Figure 8.1: Weighted percentage match at the word level computed w.r.t. the best basephone sequence. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

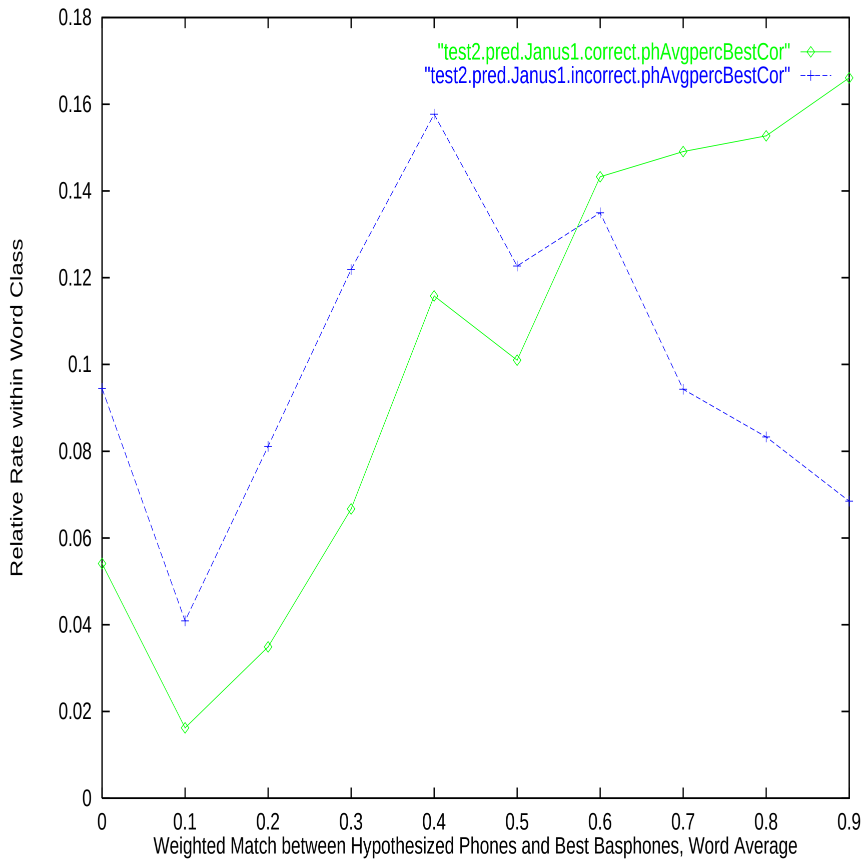


Figure 8.2: Weighted percentage match at the phone level computed w.r.t. the best basephone sequence, averaged within word. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

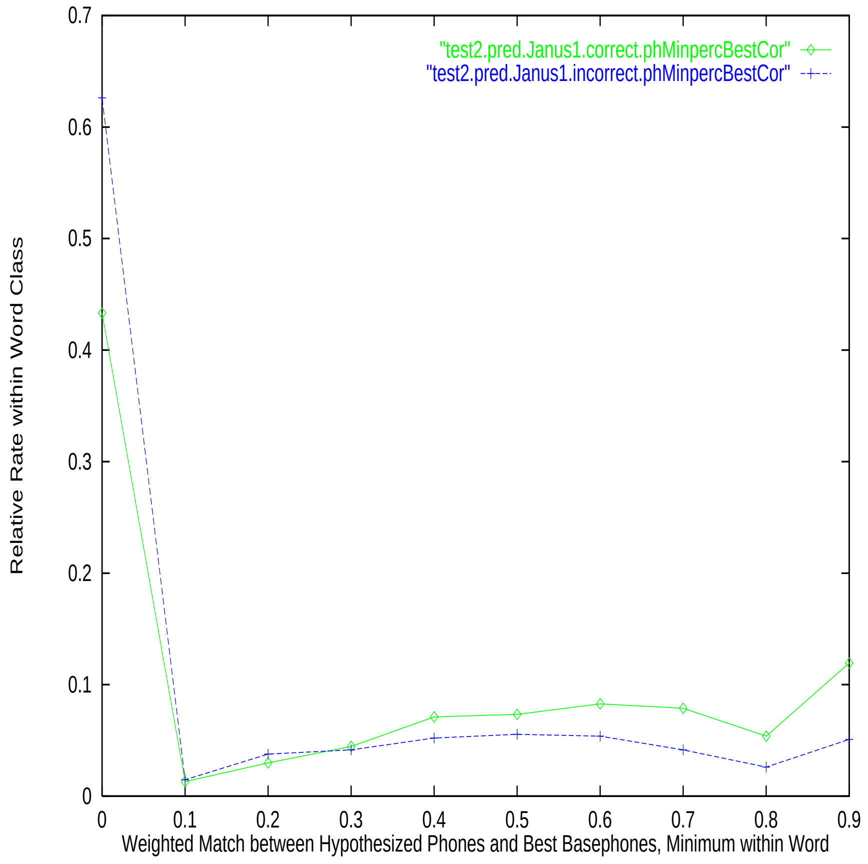


Figure 8.3: Weighted percentage match at the phone level computed w.r.t. best basephone sequence, minimum score within word. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

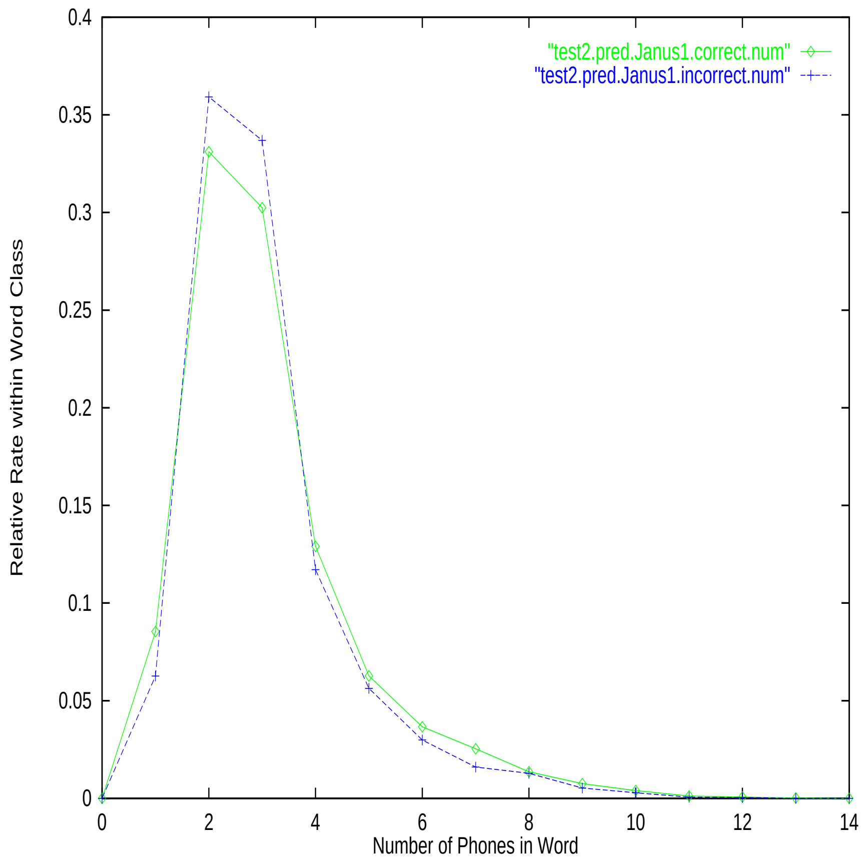


Figure 8.4: Values of number of phones per word broken out by word classes {*correct, incorrect*} *in Switchboard development data.*

In the NAB experiments the numphones predictor showed a clear separation only between the class *incorrect/OOV* and the other two other classes. The same effect is seen here, in Figure 8.4. In fact, comparing the *correct* and *incorrect* curves on this figure with those in Figure 6.1, the graphs appear to be almost identical. Each has a small preference for *incorrect* words at word lengths 2 to 4, but very little else differentiates the classes.

The distribution of N-best related scores shown in Figure 8.5 also looks extremely similar to the counterpart NAB Figure 6.3. It appears that this measure will be useful in a variety of applications. Figure 8.6 indicates that the difference between the word-level calculation of the score and the phone-level calculation of the score retains some information. Thus both word- and phone-level versions are used in the construction of the best predictor, which is described below.

In comparing the graph for avgWF in Figure 8.7 with its counterpart in the NAB experiment, Figure 6.4, we again see almost identical results. This is further corroboration that the methods we are using to measure the amount of confusion present in an N-best of list 150 is robust across application domains and systems.

Comparing the Switchboard Figure 8.8 with the NAB Figure 6.5 shows that language model score is another example of a predictor that mainly separates class *incorrect/OOV* from other classes. Comparing the *correct* and *incorrect* curves on these figures indicates very similar behavior when class *incorrect/OOV* is not considered.

Switchboard Figures 8.9 through 8.11 can be compared with the NAB Figure 6.7. Although the value of these acoustic normalization predictors are reported in the Switchboard experiment in relative difference terms instead of absolute, it is clear that the predictor is behaving similarly on each type of data. The clear separation at the left of Figure 8.11 also indicates that there is good value in including the phone-level version of this predictor.

To understand how well the H_{WC} distance metric is performing in this domain in comparison with the NAB task, we can compare Figures 8.12 and 8.13 with Figure 6.8. The effects seem very similar, with the improvement seen in this domain arising from a combination of the advantages of both the use of smoothing in the best basephone sequence and the phone-level calculation of the distance values. It appears that the H_{WC} metric is not sensitive to switching domains.

The phHWCNBestMax predictor shown in Figure 8.14 indicates that measuring H_{WC} values across the entire N-best list at the phone level, then

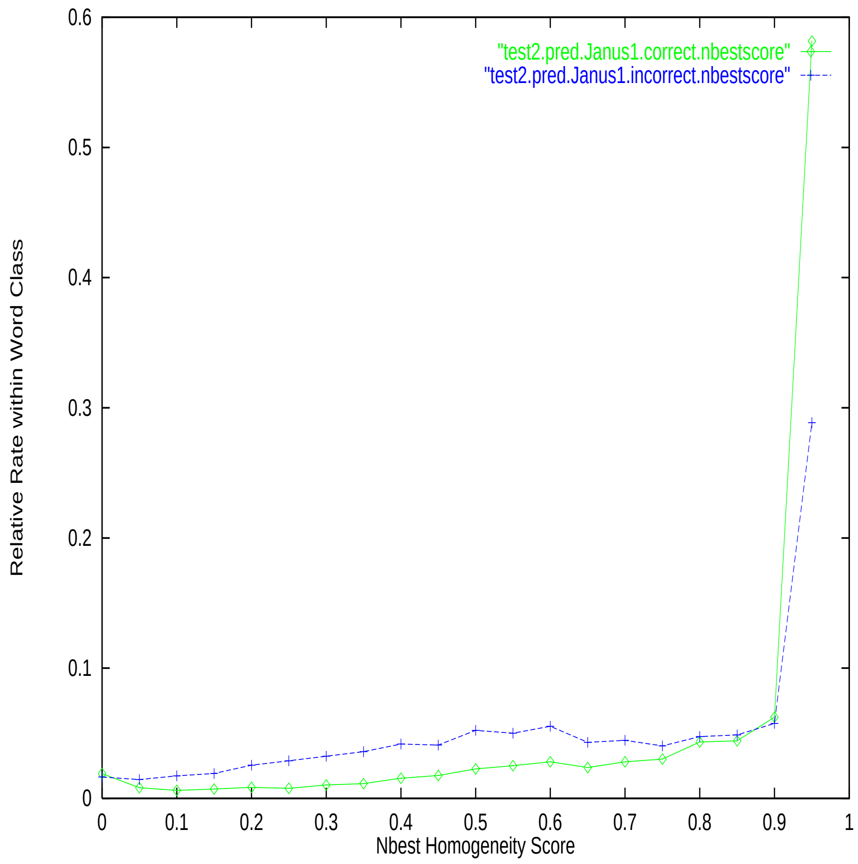


Figure 8.5: N-best homogeneity scores for words of SWB devtest in {*correct*, *incorrect*}.

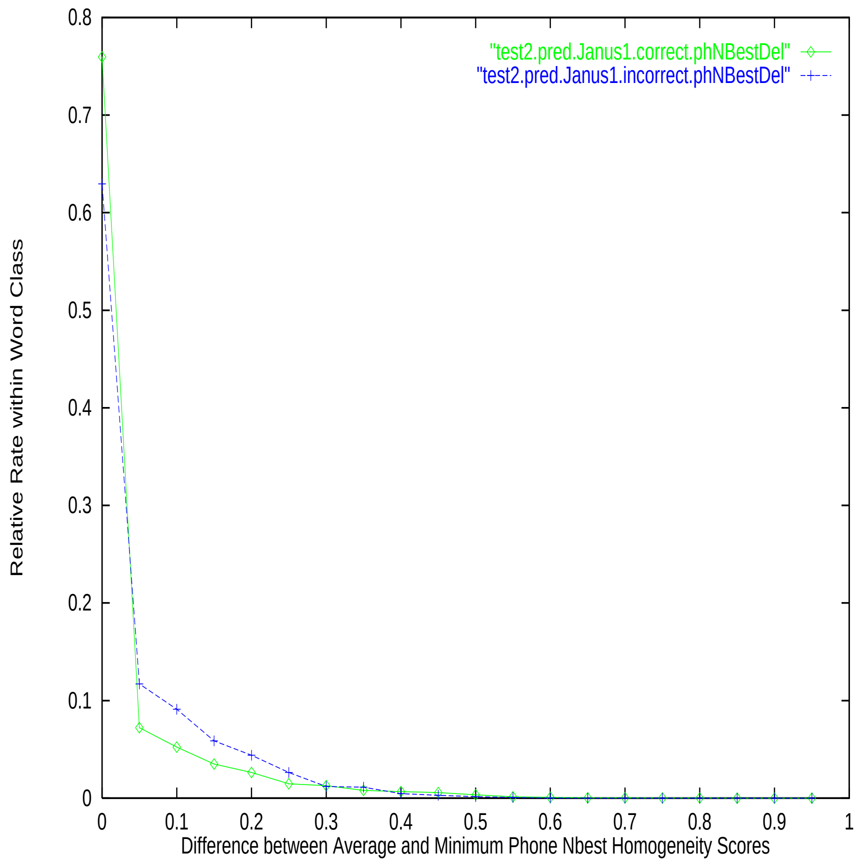


Figure 8.6: N-best homogeneity scores calculated at the phone level: Difference between average by phone and minimum phone value for words of SWB devtest in $\{$*correct, incorrect*$\}$.

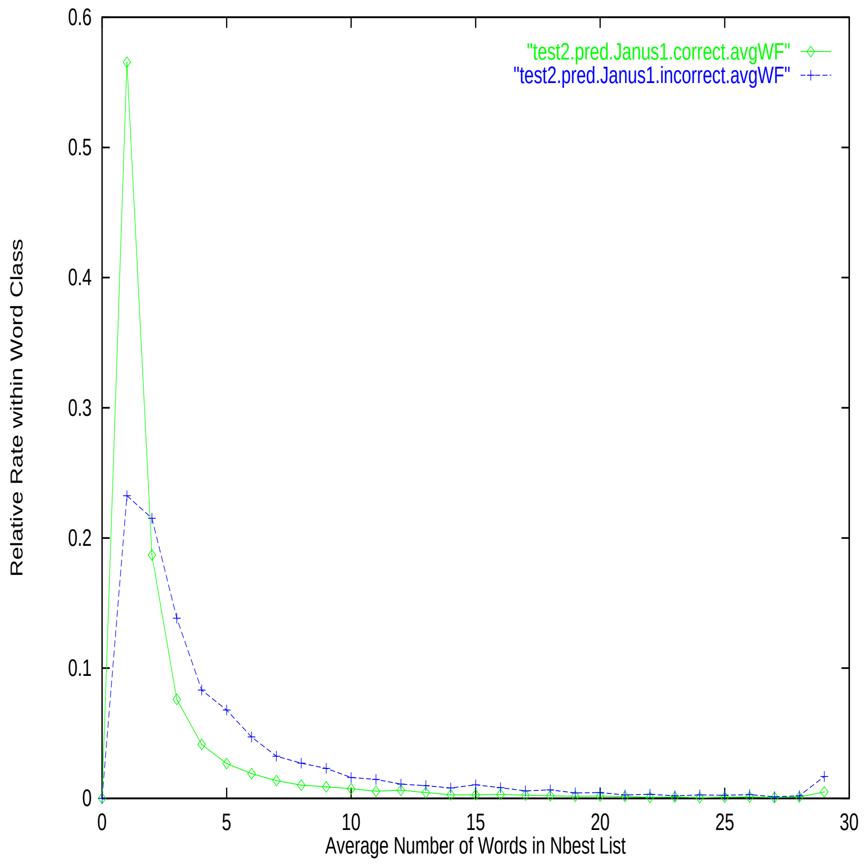


Figure 8.7: N-best average word frequency for words of SWB devtest in $\{correct, incorrect\}$.

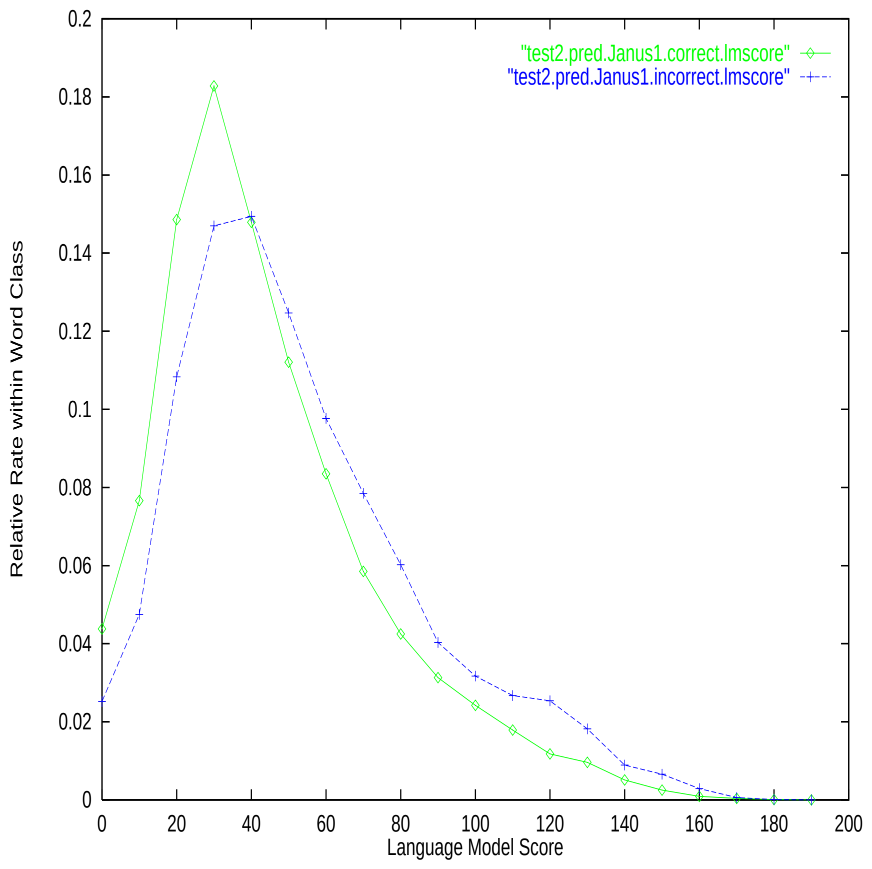


Figure 8.8: Language model scores for words of SWB devtest in {*correct, incorrect*}. Values close to zero indicate more likely word predictions. Values close to 200 indicate unlikely word predictions.

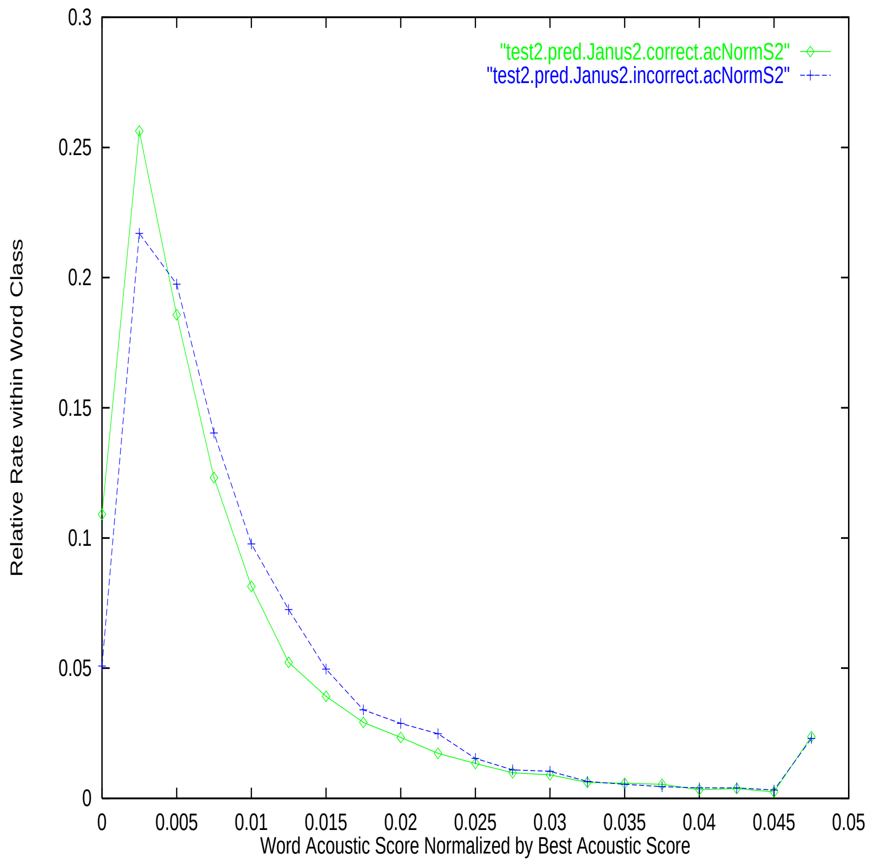


Figure 8.9: Best-acoustic-score-normalized acoustic word score curves for each class label in $\{correct, incorrect\}$ from the SWB development data.

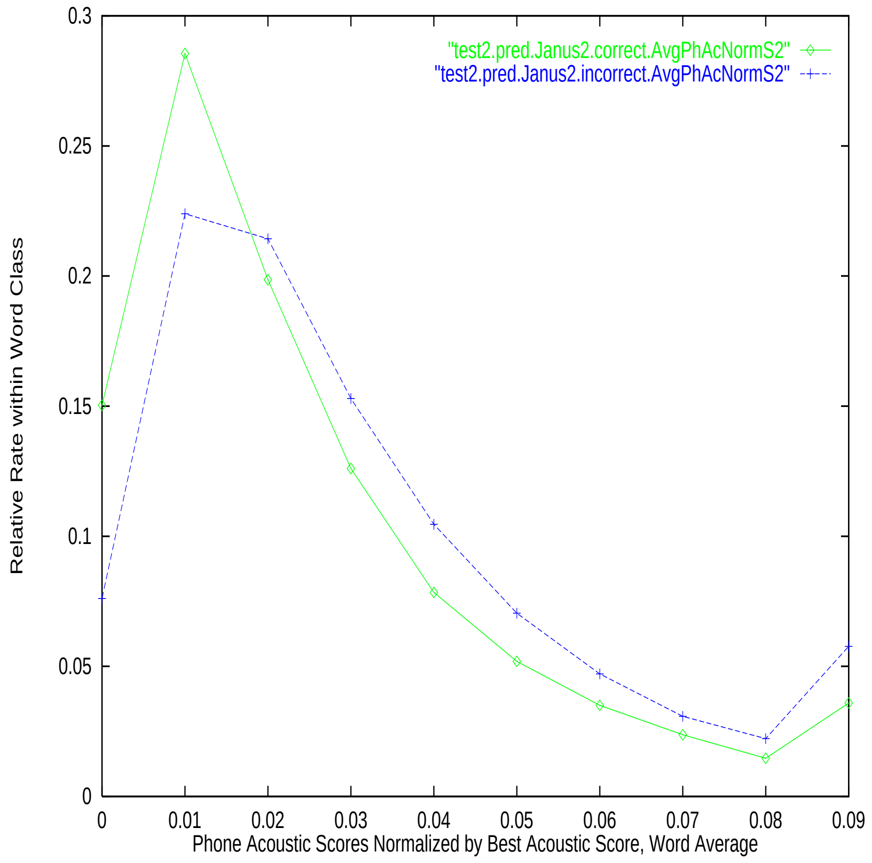


Figure 8.10: Best-senone-score-normalized acoustic phone score curves, averaged within word, for each class label in $\{correct, incorrect\}$ from the SWB development data.

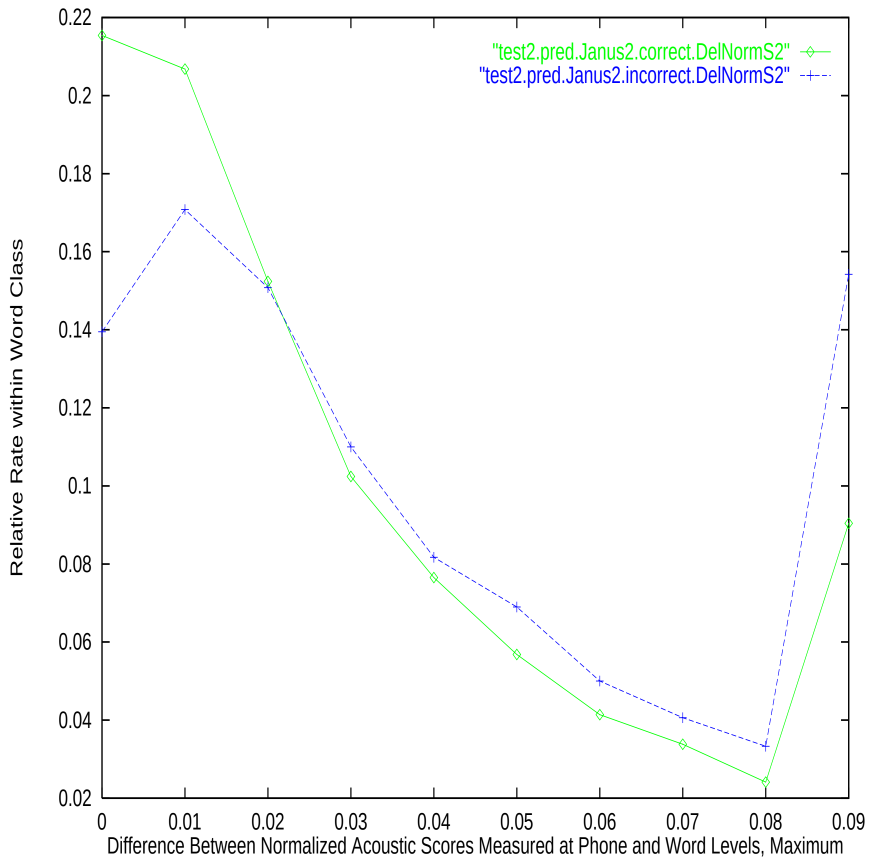


Figure 8.11: Normalized acoustic phone score curves: Difference at word level between phone maximum difference and the word-level value. For each class label in $\{correct, incorrect\}$ from the SWB development data.

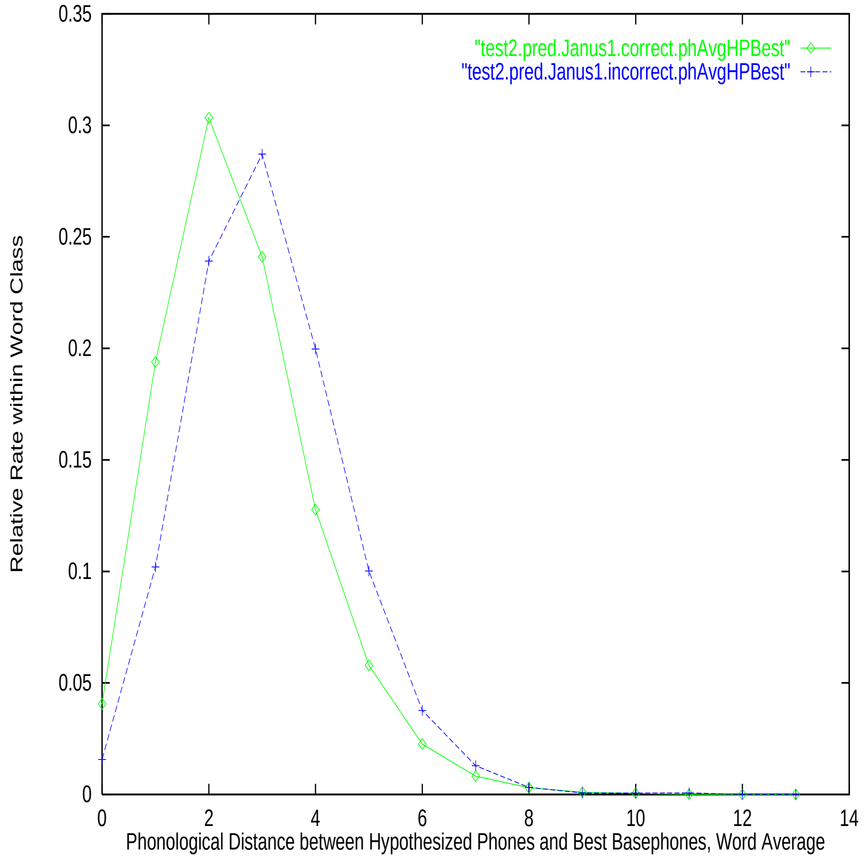


Figure 8.12: Phone-averaged H_{WC} scores computed w.r.t. the best basephone sequence for each class label in $\{correct, incorrect\}$ from the SWB development data.

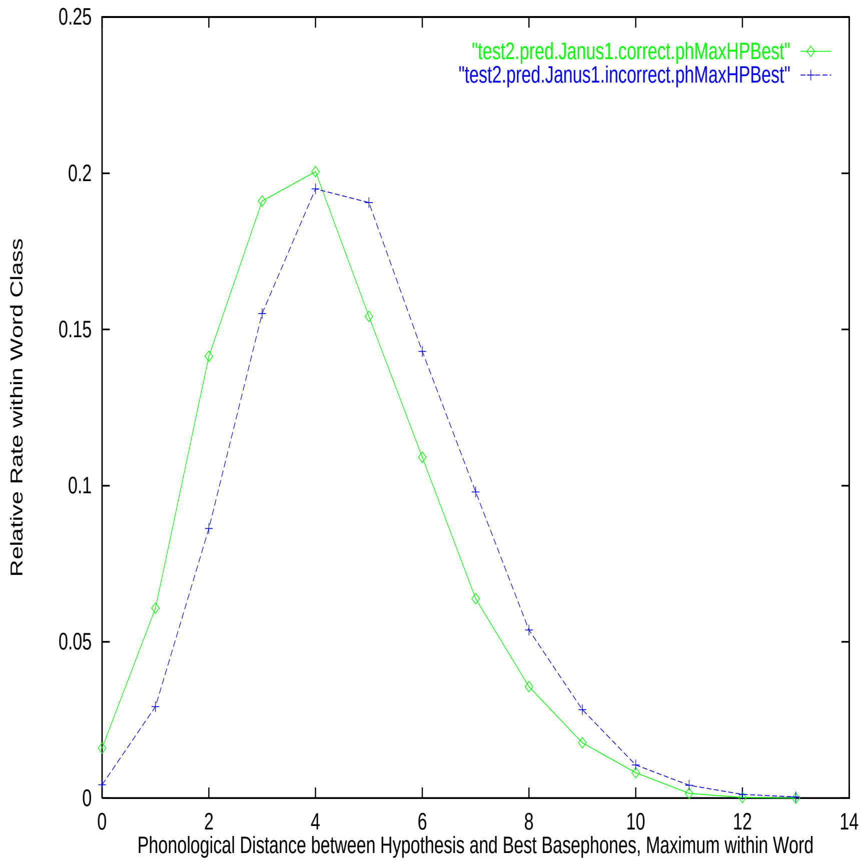


Figure 8.13: Maximum phone-level H_{WC} value computed on best basephones. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

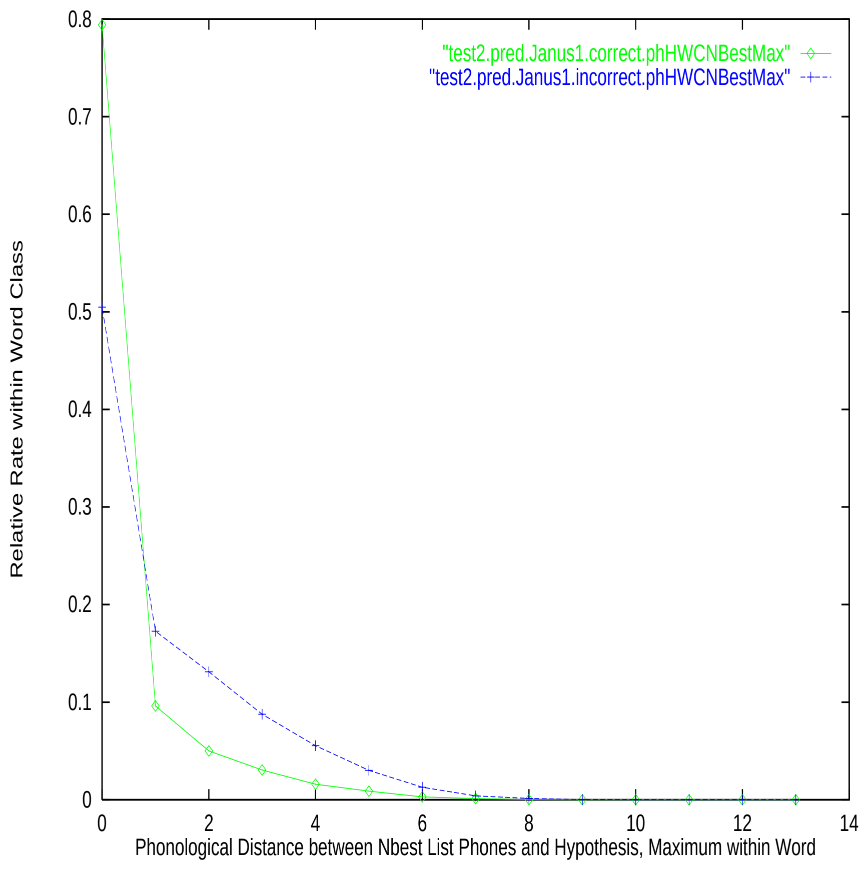


Figure 8.14: Comparison of all entries in the N-best list at the phone level under the H_{WC} metric with the hypothesized phones. Maximum average distance within word. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

looking for the maximum differences between this measure at the phone level and the word level can indicate trouble spots within words. (This predictor was not used in the NAB word-level experiments.)

Figure 8.15 indicates that the "squeezing in" of minimally short three-frame phones into hypotheses does not appear to be related with error production. (Looking for "shorties" in reference transcript alignments can help with blame assignment, however. See Chapter 9.)

The "language model jitter" predictor shows itself to be a strong separator of classes in Figure 8.16. Used on its own as a confidence measure (see Section 8.3.3), it has a very large missed error (ME) rate. Used as a predictor variable in the decision tree scheme, however, this weakness can be overcome through interactions with other variables, as described next.

8.2 Performance of the Predictors (ν)

In this section we will explore the error-predictive power available from various groups of predictor variables, as well as the interactions between them. Decision trees are trained and the predictions are made for the two classes *correct* and *incorrect*. As in the phone confidence experiment, we need only look at the the overall performance in confidence annotation for class *correct*. (This figure will be identical to the performance on either class.) Table 8.1 summarizes the relative performance of various groups of predictor variables.

The Switchboard incremental results are presented in Table 8.1.[4] The same results are presented graphically in Figure 8.17. Comparing the Switchboard results in Table 8.1 with the NAB results in Table 6.2, we see a number of interesting relationships.

The language model score gives the exact same (small) improvement in cross-entropy reduction on class *correct* in both cases. This is not surprising given how similar the separation graphs of the two classes *correct* and *incorrect* appeared to be upon visual inspection, and that the most important effect of language model score in the NAB experiment was the prediction of the presence of OOV words, which are not modeled here.

The number of times a word was seen in the acoustic training data has absolutely no effect in the Switchboard experiment, whereas in the NAB experiment it had relatively big effect on the prediction of OOV words in the input.

[4]See Section 8.3.2 for a discussion of the entry labeled "nbest+lmscore+acNormS+...".

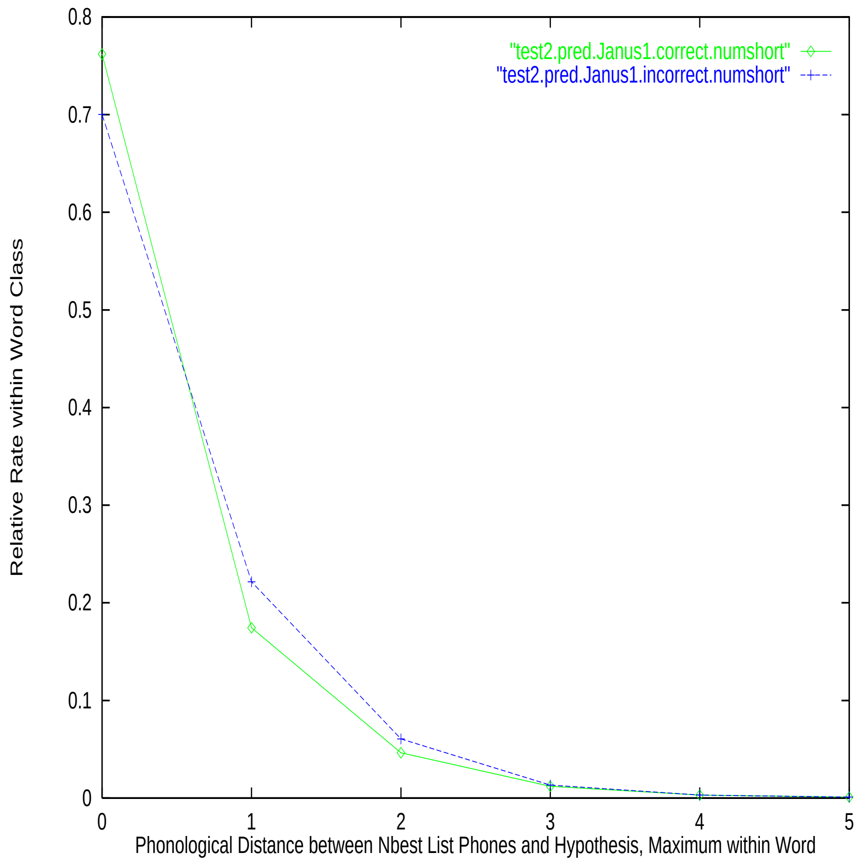


Figure 8.15: Number of minimum-length phones in hypothesized word. Shown for each class label in $\{correct, incorrect\}$ from the SWB development data.

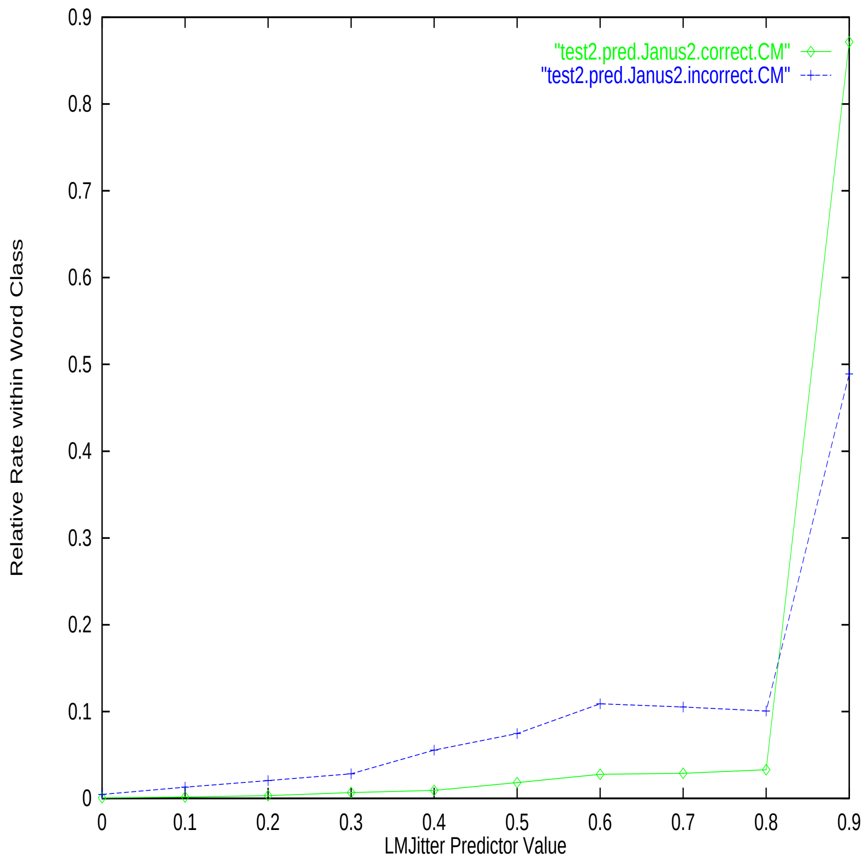


Figure 8.16: Language model jitter as a predictor variable. Shown for each class in {*correct, incorrect*} from the SWB development data.

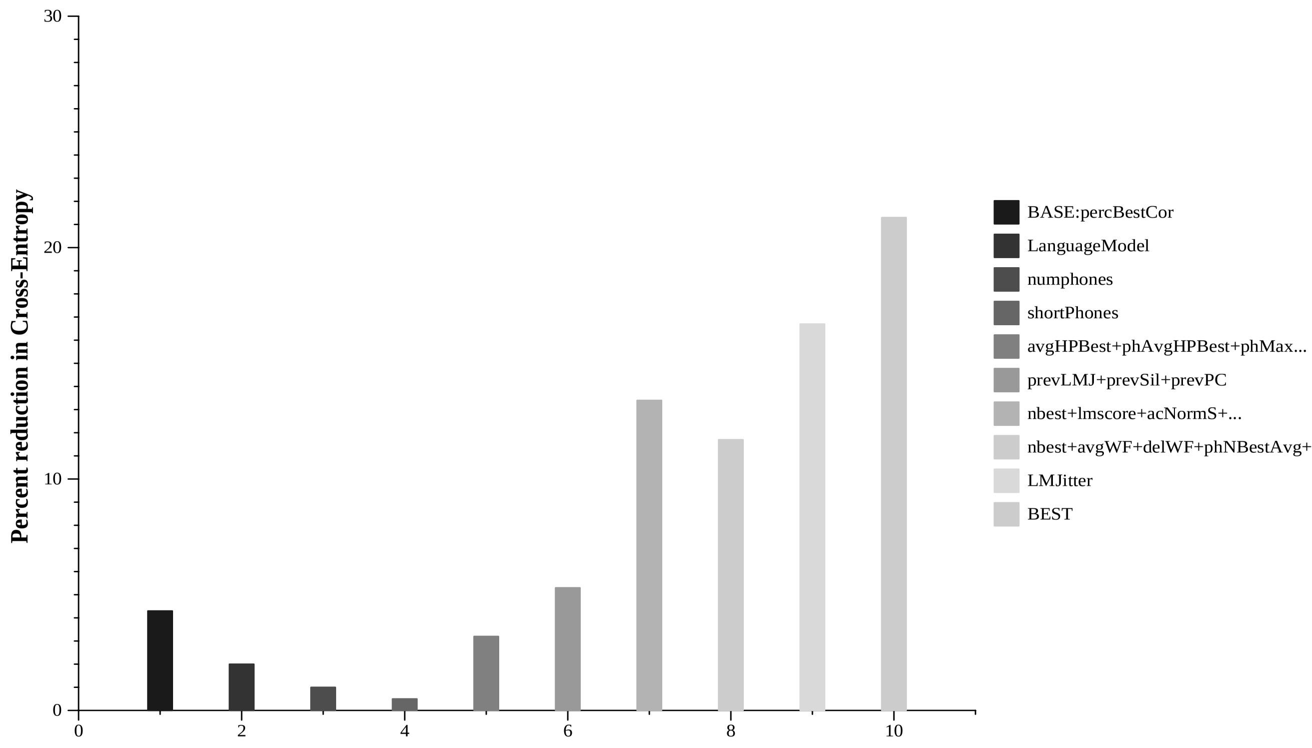


Figure 8.17: Incremental and confidence annotation results compared to BASEline and BEST systems.

Predictor Variables	%reduction Cross-entropy
BASE:percBestCor	4.3%
LMscore	2.0%
numphones	0.5%
numShort	0.5%
acNormS+AvgPhAcNormS+DelNormS	4.2%
avgHPBest+phAvgHPBest+phMax...	3.2%
prevSil+prevPC	5.3%
nbest+lmscore+acNormS+...	13.4%
nbest+avgWF+delWF+phNBestAvg	11.7%
LMJitter	16.7%
BEST	21.3%

Table 8.1: Improvement in 2-class word confidence annotation as a function of predictor variables used. Performance measured in percent reduction in cross-entropy on SWB.

The numphones predictor has similar (and also small) effects in both cases.

The Switchboard BASE:percBestCor and the NAB BASE1:percPhAll baselines appear to be similarly powerful. In fact, the Switchboard baseline is not exactly comparable to the NAB BASE1 system, since it relies on the best basephone sequence and not an independent phone-only decoding as its reference. But the baseline results do give us a good feel for how well we can expect to do if all we do is look at some external source of phone sequences. In both cases using the H_{WC} metric instead of a frame-level match yields a minor improvement.

The effects of the acoustic normalization predictors (acNormS, AvgPhAcNormS, and DelNormS) are more powerful in the Switchboard experiment than in the NAB (acNormS only). This is due to the increased effectiveness of making the calculation at the phone level and then propagating the information upward.

In both systems the Nbest predictors (nbest, phNBestAvg, avgWF, and delWF) yield very similar predictive power.

In both cases a similar predictive power is gained from looking at the likelihood that the previous word was *correct* (prevPC).

Finally, the best single predictor in the Switchboard experiment is the LMJitter variable. In the next section we use all of the predictor variables together to build and evaluate our best system. In analyzing that best predictor we will see that LMJitter plays a lead role.

8.3 Performance of the Best Annotator (ν)

In this section we evaluate the best confidence annotator constructed from the predictor variables presented in the previous section.

8.3.1 Domain-independent Evaluation (ν)

The top branches of this tree are shown in Figure 8.18. The first test applied by this annotator is on the value of LMJitter. The threshold is 0.99, which means that the question is, "Did this word drop out under even one pair of LW/IP values?". If so then the probability that the word is *correct* drops dramatically.

If not, then the next question is "Was there any sign of confusion in the N-best list?". This is measured using the avgWF variable, which is thresholded just below the value 2. The likelihood that the word in question is *correct* is very high if both the LMJitter and avgWF tests indicate no doubt and no confusion. After those two tests the words in this tree path are further checked out by match comparison with the best basephone sequence. Relatively minor adjustments up and down are made as a results of that test.

For those (suspect) words which do not remain completely and steadfastly present under the LMJitter variations, the next test is comparison against the best basephone sequence. Those words found acceptable after this test have their $P(C)$ values refined again under the LMJitter test at a lower threshold. Those found lacking in comparison with the best basephone sequence have their $P(C)$ values dropped and are then tested looking at the value of H_{WC} across the phones found in the N-best list. (Recall that this is another measure of confusion in the N-best list.)

After these initial tests, the additional variables found in the best tree include phNBestAvg, the acNormS predictors, prevPC, and lmscore.

Figures 8.19 and 8.20 show the $P(C)$ curves created by the best annotator on word classes *correct* and *incorrect*. Even though the NAB and Switchboard experiments yield very similar results in terms of the cross-entropy

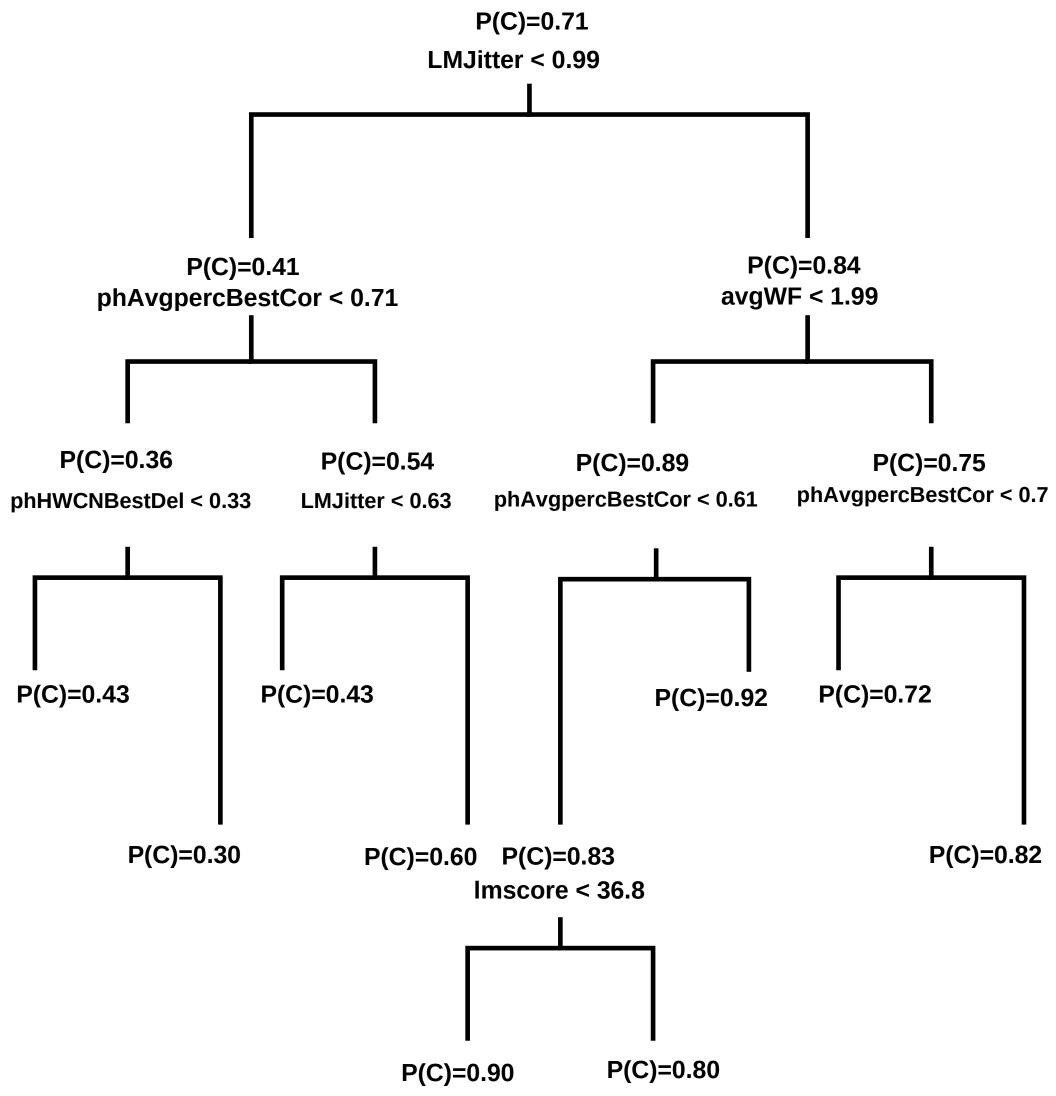


Figure 8.18: Top branches of the best word-level predictor of error labels {*correct, incorrect*}. Switchboard data.

reduction figures, the curves in Figure 8.20 compare favorably to those in Figure 6.15. In the Switchboard curves show that in comparison there is a much clearer region in which *incorrect* words are receiving clearly lower $P(C)$ labelings. This is born out by comparison of the related False Alarm/Missed error curve for Switchboard (Figure 8.22) with the analagous NAB curve (Figure 6.16). In the Switchboard case there is a clear region in the middle range of potential thresholds in which the sum of False Alarms and Missed Errors is smaller than the original error rate. This is just barely true for the NAB curves.

8.3.2 Alternative Learning Mechanisms (ν)

Decision trees suffer as a learning mechanism because with each split made in the tree the training data becomes more fragmented. This is not a learning method in which each training data point can influence the construction of decision boundaries. The main benefits of the decision tree approach are that:

- non-linearities in the relations between predictors and response fall out naturally,
- interactions between variables fall out naturally, and
- it is possible to "read" the results to get an intuitive sense for how the predictors relate to the response.

In this section I compare the decision tree approach with three other methods, each of which share some but not all of the character of decision trees.

This effort was motivated in part by some interesting but incomplete results reported by other researchers:

- The group at SRI [5] reported results using neural networks, which indicated tentatively that they outperformed decision trees in terms of reduction in cross-entropy of confidence annotations.
- Several groups [51] [20] have tried generalized linear regression techniques. This learning mechanism seemed to produce similar-quality results to the decision trees in terms of reduction in cross-entropy.

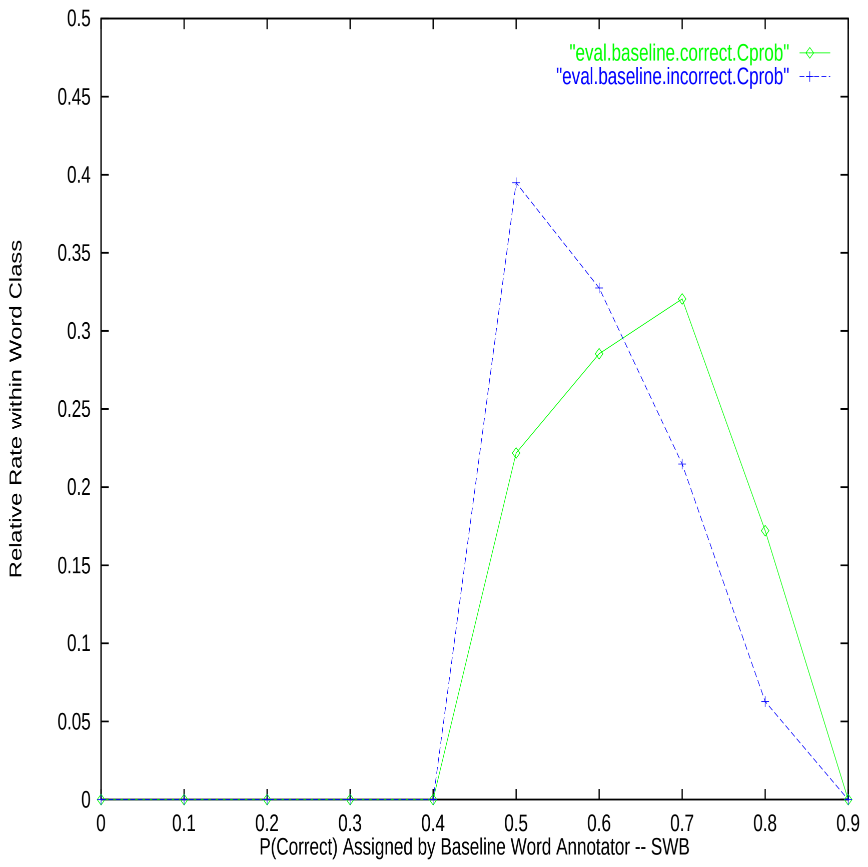


Figure 8.19: Distributions of values of $P(C)$ for the two word classes on SWB. BASEline system.

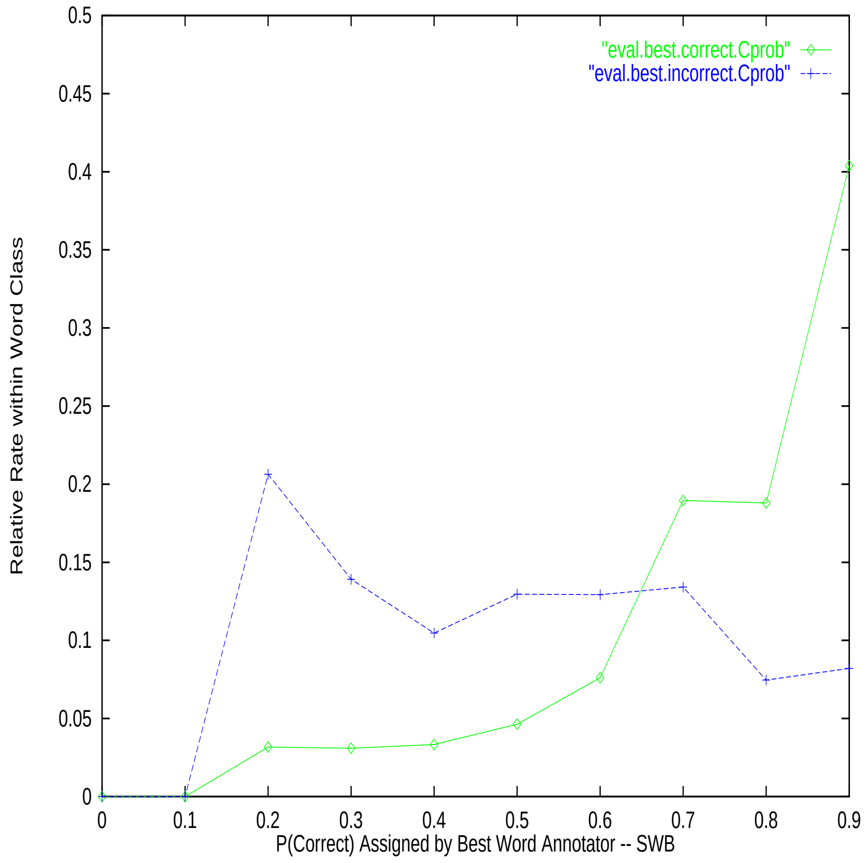


Figure 8.20: Distributions of values of $P(C)$ for the two word classes on SWB. Best system.

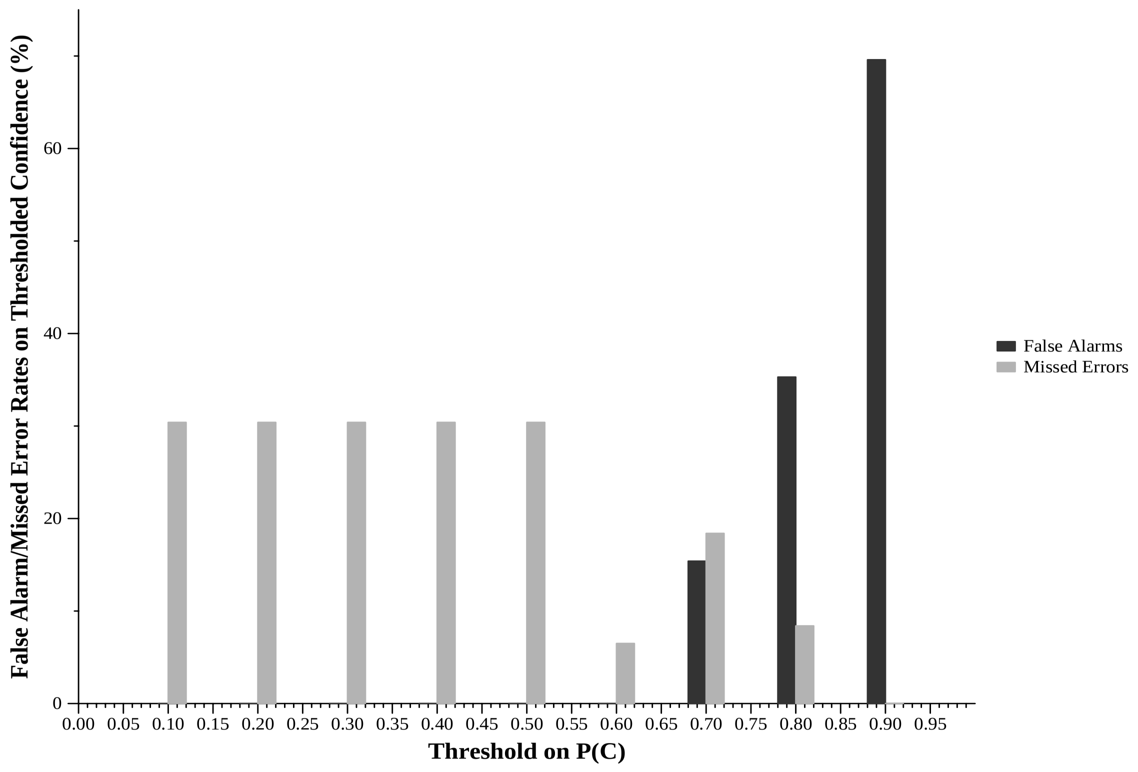


Figure 8.21: False alarm/missed error rates at various $P(C)$ thresholds for the BASEline word-level predictor of error labels in classes {*correct*, *incorrect*} on SWB data.

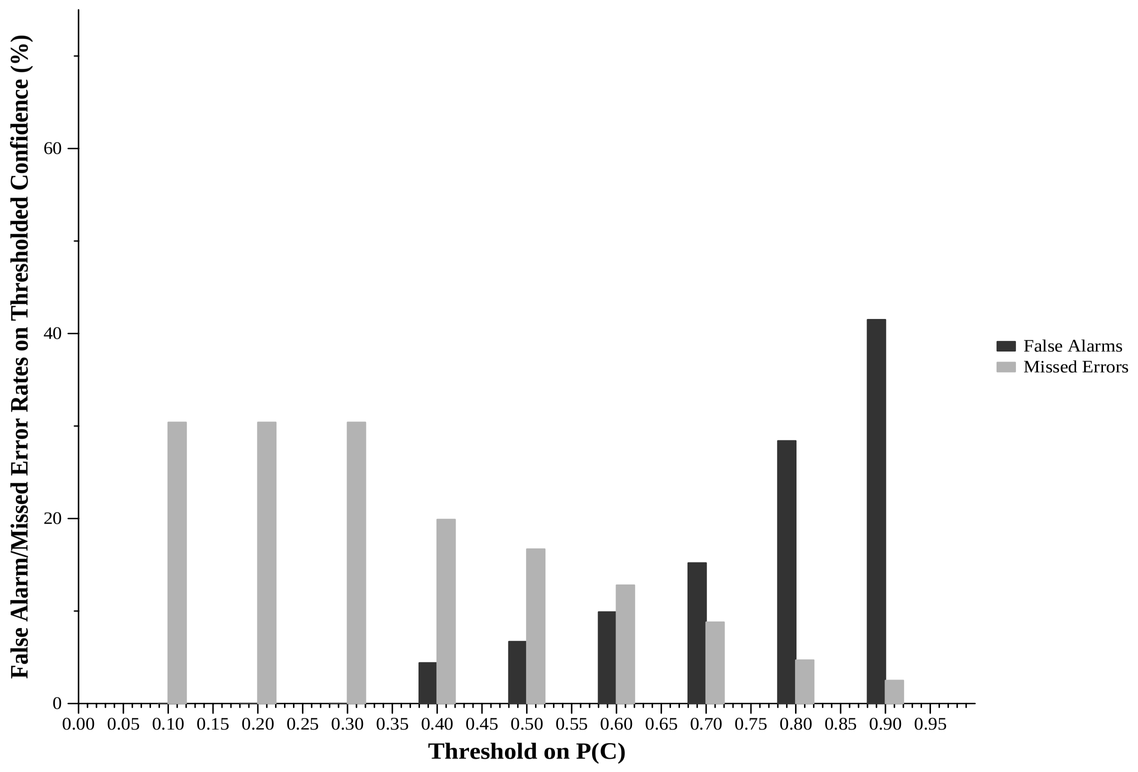


Figure 8.22: False alarm/missed error rates at various $P(C)$ thresholds for the best word-level predictor of error labels in classes {*correct, incorrect*} on SWB data.

The results reported next comprise a complete comparison between these methods, and includes additionally a method known as "generalized additive models". Table 8.2 includes performance figures that compare these methods on two sets of predictors. The first set, referred to here as "Previous Best", is a re-implementation of the set of predictors reported by the groups mentioned above. [5]

Predictor Variables	Decision Tree	GLM	GAM	Neural Net
Previous Best	13.4%	12.6%	14.2%	15.1%
Current Best	21.3%	23.1%	24.0%	23.8%

Table 8.2: Performance in reduction in cross-entropy, comparing two sets of predictor variables and four learning/combination methods. Switchboard data.

Each learning method is described below, followed by some concluding remarks.

Generalized Linear Models [37] [59] [7] Generalized linear models are an extension of standard linear regression techniques. In the standard technique, a numeric response variable y is influenced by predictors only through the (linear) predictor function

$$y = \beta^T x + \epsilon$$

where T is the length of a vector of regression coefficients, the set of predictors x includes explicit modeling of interactions between subsets of individual predictors, and ϵ represents an error term which is assumed to be distributed normally.

A generalization of this model is available which includes two main changes:

1. Categorical response variables are possible. In our case two levels, the classes *correct* and *incorrect*, are modeled, with $y \in [0, 1]$.

2. The variance of the response y is modeled to change with the mean μ of y.

[5]These have been described in detail previously in this chapter. The exact list is lmscore + numphones + nbest + avgWF + delWF + numShort + acNormS + duration. The second set, referred to as "Current Best", includes all predictors reported here.

Binary response values, such as our response variable *class*, can be modeled with the assumption that they have a mean $\mu \in [0,1]$ and a variance $\mu(1-\mu)$ that changes with the mean. Generalized linear models extend the original linear regression model with two changes:

1. A *link* function is used to describes how the mean μ varies with the predictors : $g(\mu) = \beta^T x$, and
2. A *variance* function is used to capture how the variance of y changes with its mean μ: $\mathrm{var}(y) = \phi V(\mu)$, where ϕ is a constant.

The final form of the generalized linear regression model is thus:

$$\mu = f(\beta^T x)$$

where f is the inverse link $f = g^{-1}$.

If we use the link function called "logit",

$$\mu = \frac{e^\eta}{1 + e^\eta}$$

we are guaranteed that the mean μ will be in the interval $[0,1]$, which is appropriate for modeling a probability that varies between two possible outcomes. This logit link, together with the binomial variance function $V(\mu) = \mu(1-\mu)$ is used to define the regression model. [6]

Thus the model can be read as meaning "the logit of the mean of the response variable is the linear regression ..."

$$\mathrm{logit}(\mu) = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots \beta_T x_T$$

This model does allow each data point to influence the final model directly. It models only linear relationships between predictor and response. Interactions between predictor variables must be identified explicitly before regression is performed. It is possible to analyze regression model coefficients to understand the contributions made by the various predictors, but the result is not quite as useful as the "reading" of decision trees.

[6]In all of the experiments reported here in which a binomial link function is used (including the generalized additive models), the logit link was compared with two other link functions, commonly called "probit" and "complementary log-log" [7] [59]. In all cases the probit link function performed identically with the logit function and the complementary log-log function performed slightly worse than logit and probit. Thus only logit-based results are presented.

Generalized Additive Models [59] [7] We can extend the generalized linear model to capture nonlinearities by replacing the linear predictors with a parameterized set of additive predictors. If each response observation is taken to be a random variable Y_i which is distributed as a binomial with parameters (n_i, p_i), then the p_i is determined by

$$\text{logit}(p) = \alpha + \sum_{j=1}^{p} f_j(X_j)$$

In our case we take the functions f to be the family of cubic B-splines. In effect this approach fits a cubic B-spline curve to the experienced values of each predictor variable[7] in turn, then perform the generalized linear prediction described in the previous section, including the use of the link function.

This model does allow each data point to influence the final model directly. It also models nonlinearities, as long as those nonlinearities can be captured by the cubic B-spline model and the addition of such terms. Interactions between predictor variables must be modeled explicitly before regression is performed. "Reading" generalized additive models is possible (by examining learned regression coefficients, etc.), but not quite as straightforward as looking at decision trees.

Neural Nets For the neural net model, I used a standard multi-layer perceptron [25] with one input node per predictor variable, twice as many hidden nodes as predictors, and one output node. Sigmoid activation functions were used at both hidden and output nodes. Inputs were not normalized in any way. The standard back-propagation algorithm was used, but the optimization function was entropy-based, not the standard least-squares method [59].

This approach allows each data point to influence the final model directly, captures nonlinearities directly, and models interactions directly. It is not usually possible to "read" a neural net after it has been trained, however.

Summary As we can see from Table 8.2, performance in confidence annotation improves as we allow each data point to influence the outcome. This factor combines with with the ability to capture nonlinearities in the relation between predictors and response to create clear increases in performance. If

[7]We model interactions between variables explicitly, and treat them as individual terms.

we want to achieve both the best performance and "readability" of results, we can use both decision trees and either GAM or neural net models in parallel.

8.3.3 Application-specific Evaluation (ν)

In this section we will revisit the idea (presented originally in Section 6.9.2) that it is important to analyze confidence annotators in an application-specific manner in order to understand how useful they really are or aren't. To do this we will look at two examples of application-specific measures of the usefulness of confidence annotations.

The first is analagous functionally to the recognition output filtering application presented on the NAB data in Section 6.9.2. Here the filtering is applied with a different goal in mind, however. Instead of tagging multi-media databases we are interested in finding specific speakers or conversations for which the recognizer is having a hard time.

The second application discussed is the use of annotated recognizer output for the filtering of data used in unsupervised acoustic adaptation.

In both cases we find that although the cross-entropy figure of merit is a good indicator of whether the annotator is working or not, that it is not necessarily a good predictor of exactly how useful the annotations will be in any given application.

Using Confidence Annotators to Filter Speech Recognition The first, another filtering example, is analagous to the discussion in Section 6.9.2 in which filtered recognizer output was set up for possible use with multi-media retrieval applications.

In this case we apply exactly the same filtering approach to produce the figures presented in Tables 8.3 and 8.4. These tables contain the analagous figures to the NAB results presented previously in Tables 6.6 and 6.7.

Comparing the threshold at which roughly 30% of the data makes it through the filter, we see that the WER in the filtered data is higher in the Switchboard case – about 10% as opposed to 3%. This is about what we might expect given that the base error rate in Switchboard is about three times that in the NAB experiments. An alternate view of this filtering behavior is presented in Figure 8.23. The upper curve in the figure represents the amount of data passing through the filter. The lower curve represents the quality of the data that passes through. We can see from this figure that at lower threshold values on $P(Correct)$ the tradeoff between these two is relatively even, but that at higher thresholds the amount of data that passes the filter drops more sharply in comparison with gains made in the quality of the filtered data. A comparison of this figure with the analagous figure for the NAB data, Figure 6.19, indicates that both curves are in more extreme positions for the NAB data than for the SWB data. Thus, although both word-level annotators that we have constructed (NAB and Switchboard) yield almost exactly the same reduction in cross-entropy, the end effect on the filtering process is closely tied to the base error rate of the system. Thus in evaluating how much a confidence annotator can help with this sort of task it is important to keep in mind the overall difficulty of the task.

In a related application, we can apply filtering at the conversation side level. This means that we will not filter individual words out of the stream, but rather entire conversation sides. Figure 8.24 displays one data point per conversation side (of independent test data). The vertical axis indicates the average confidence annotation of all words in the conversation side. The horizontal axis indicates the actual word error rate (WER) of the conversation side. We see from this figure that the confidence annotations averaged in this manner can indeed identify "problem cases" in which some combination of speaker, telephone handset, and line acoustics is causing the recognizer problems.

Thus filtering at various levels proves to be a natural use of confidence annotations.

P(C) Thresh	Avg WER	%age Above Thresh
0.5	27%	63%
0.6	23%	60%
0.7	16%	54%
0.8	12%	41%
0.85	10%	37%
0.9	9%	28%
0.95	5%	13%

Table 8.3: Word confidence annotations can be used to filter data for regions of high recognition quality (low WER). Switchboard data.

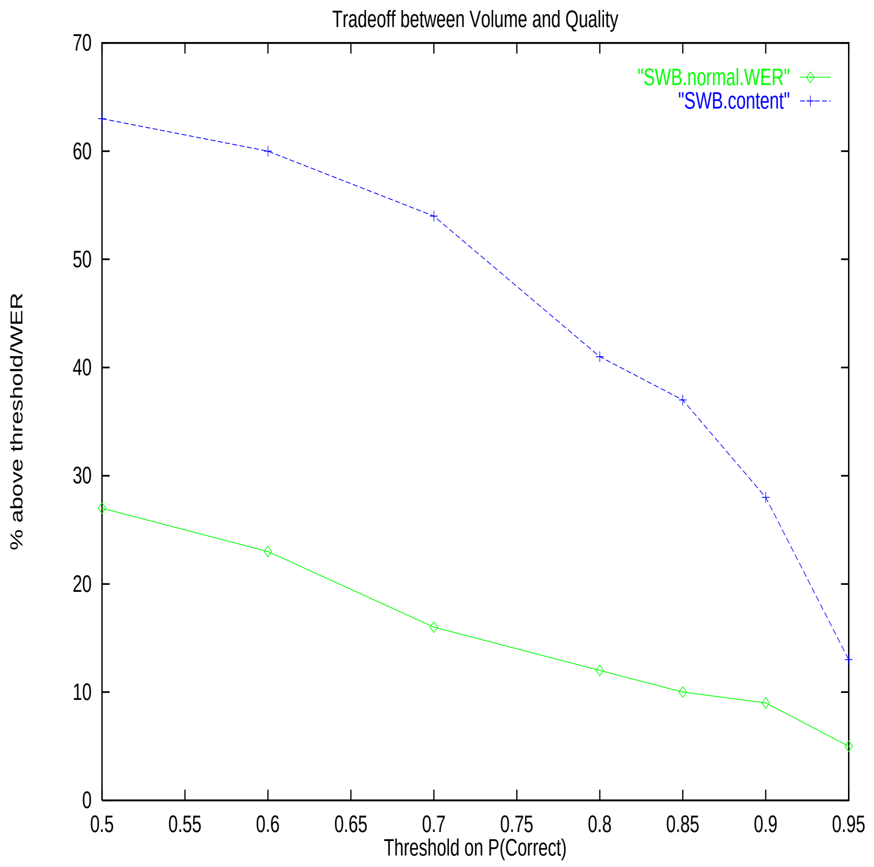


Figure 8.23: Tradeoff between quantity and quality of data passing through filter of $P(Correct)$. Switchboard data.

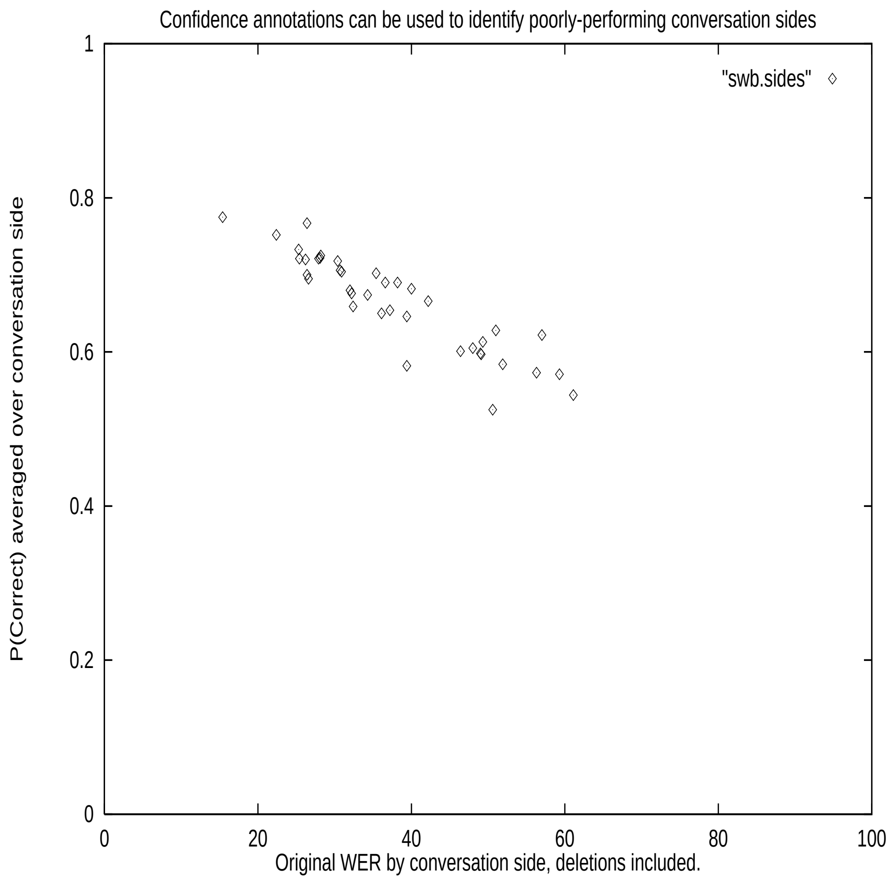


Figure 8.24: Averaging the confidence annotations within conversation side identifies problem situations. Switchboard data.

P(C) Thresh	Avg WER	%age Below Thresh
0.3	73%	23%
0.4	69%	25%
0.5	65%	27%
0.6	61%	30%
0.7	54%	46%
0.8	39%	59%
0.85	36%	63%
0.9	31%	72%
0.95	27%	87%

Table 8.4: Word confidence annotations can be used to filter data for regions of low recognition quality (high WER). Switchboard data.

Using Confidence Annotators to Improve Acoustic Adaptation Another use of confidence annotators is in the improvement of unsupervised acoustic adaptation.

Like many state-of-the-art speech recognizers, the Janus system is configured to optionally use MLLR adaptation for its acoustic models [35]. This adaptation approach requires a transcript against which the acoustic models are aligned before a linear regression is run to effect the adaptation. For supervised adaptation the transcript can simply be the reference transcript, but for unsupervised adaptation a recognition hypothesis is required.

The simplest approach to using a recognition hypothesis for MLLR adaptation is to run the recognizer once, creating a recognition hypothesis, which is then used as the reference for adaptation. Another option is to use a confidence annotator to weight the contribution of various portions of the hypothesis used with MLLR. For example, a word labeled with a $P(C)$ of 0.2 would have only a 20% contribution to the adaptation regression, whereas a word labeled with a $P(C)$ of 0.9 would contribute to the adaptation with a weight of 90%. Yet another option is to threshold the confidence annotator, thus making a binary decision about whether a hypothesized word will contribute to the adaptation calculation or not.

To illustrate the relationship between the reduction cross-entropy values for various confidence annotators and their effects on acoustic adaptation, we ran the following experiment.

First, two confidence annotators were produced:

1. *LMJitter confidence annotator:* The LMJitter predictor variable described above was used without transformation via decision trees. This was the rudimentary confidence annotation approach used with Janus before the development of this approach. The 0 cross-entropy created by this unsmoothed predictor is 12.1%, as compared to 16.7% after tree-based smoothing.

2. *Best confidence annotator:* The best decision tree-based confidence annotator described in the previous section (see Figure 8.18) was used.

As reported above the "Best" confidence annotator performs almost twice as well as the "LMJitter" annotator under the reduction in cross-entropy measure.

The following experimental conditions applied:

1. *New Janus system*: A Janus system distinct from the version used in the previously reported experiments was used. The changes included the addition of common word phrases, the acoustic training of these word phrases, and adaptation of the language model scoring elements in the search process to incorporate these word phrases.

2. *Old hypotheses*: The first round of adaptation, in which the entire hypothesized output is used without any confidence annotation, was done with the original hypotheses created by the original system reported above.

3. *Old LMJitter annotations*: The adaptation runs using the LMJitter annotator used values of LMJitter created from the original lattices used in the experiments above.

4. *Old Best annotations:* The adaptation runs using the Best annotator used values of the Best annotator derived from the original experimental results above.

5. *Same length adaptation training:* All 5 minutes of each conversation side were used to train adapted acoustic models.

6. *Shorter test*: Instead of testing on the complete 5 minutes of each evaluation test conversation side, only the first 30 seconds of each conversation side were used.

In short, this means that:

- The two confidence annotators used to filter data for adaptation were both developed using the original Janus system and data.
- The acoustic adaptation was trained on the complete test set.
- Test recognition runs using adapted acoustic models were done using an improved version of Janus on shorter tests.
- Due to the use of a different Janus system, no unadapted performance figures (WER) that are comparable to the adaptation figures are available for comparison.

On the shorter test, after adaptation based on complete (unfiltered) hypotheses, the WER was 31.5%. After application of the LMJitter confidence annotator, using it in the weighted fashion, the results were 31.1%. After application of the LMJitter confidence annotator, using it in the thresholded manner with a test of $LMJitter < 1.0$, the results were 30.7%.

These adaptation results are summarized in Table 8.5.

Method of Hypothesis Use	Resulting WER under MLLR
Complete HYP	31.5%
Weighted LMJitter	31.1%
Thresholded LMJitter	30.7%
Weighted Best SWB	31.0%
Thresholded Best SWB	30.7%

Table 8.5: Effects of using confidence-annotated hypotheses for unsupervised MLLR adaptation. Switchboard data.

Despite the fact that the best confidence annotator clearly outperforms the LMJitter-based annotator in terms of reduction in cross-entropy, no differences are observed in the "better" system's ability to assist with unsupervised acoustic adaptation! This is a very clear example of the need to test confidence annotators in a task-specific context.

$P(C)$ Threshold	WER under MLLR	Amount of data
0.50	31.2%	63%
0.65	30.8%	57%
0.70	30.7%	54%
0.80	31.1%	41%

Table 8.6: Trading off amount of data against quality of data for unsupervised MLLR adaptation . Switchboard data, best confidence annotator.

8.4 Summary

In this chapter we replicated the confidence annotation method developed for read speech in Chapter 6 in the new domain of conversational telephone speech. To perform these experiments we switched from the Sphinx-II recognizer to the Janus recognizer, which was developed specifically for conversational tasks. We incorporated improvements developed in Chapter 7 that indicated that many predictors of error give increased performance when measured at the phone level. We also employed the detailed evaluation process shown first in Chapter 6, this time comparing in the performance of predictor variables across the two domains. Our best Switchboard confidence annotator separates correctly recognized words from incorrectly recognized words approximately five times better than a simple baseline system, with an overall reduction in cross-entropy for the labeling of class *correct* of 21.3%.

In the experiment reported here, the most powerful predictor variable that contributes to this best system's identification of *correct* words is called the "LMJitter" score. It is a measure of how much "push in/pop out" behavior a word will experience under varying contribution weights of the language and acoustic models. If a wide range of language weights and insertion penalties are used in searching a lattice, it is possible to count how often a particular word appears and disappears from the hypothesis with the changing LW/IP pairs. For those words which fall out of the analysis under even one set of LW/IP pairs tried, the probability that it is has not been correctly recognized increases tremendously.

For those words which do not fall out under any LW/IP pair, the next important question is "Was there any sign of confusion in the N-best list?". This is measured using the avgWF variable, which describes how many other words are present in the N-best list at the location of the word in question.

This variable is thresholded just below the value 2, which means that any confusion at all is telling. The likelihood that the word in question is correct is very high if both the LMJitter and avgWF tests indicate no doubt and no confusion.

After these first two tests the words surviving this sequence of tests are further checked out by match comparison with the best basephone sequence. Relatively minor adjustments up and down in likelihood of error are made as a results of this comparison.

For those (suspect) words which do not remain completely and steadfastly present under the LMJitter variations, the next test is also comparison against the best basephone sequence. Those words whose phone sequences match the best basephone sequence satisfactorily have their $P(correct)$ values refined again under the LMJitter test at a lower threshold.

Those found lacking in comparison with the best basephone sequence have their $P(correct)$ values dropped. These words are additionally tested using additional variables that look for phonological confusion (under the H_{WC} metric) in the N-best list.

The additional variables found in subsequent tests include factors related to acoustic score normalizations, language model scores, and the probability of correctness for preceding words.

Analysis of the *correct/incorrect* separating power of predictor variables indicates that making calculations at the phone level and then propagating them up to the word level wherever possible improves performance. This is especially true for comparisons of hypotheses against the best basephone sequence, acoustic score normalizations, and various measures of confusion in the N-best list.

Our analysis also shows that predictors formerly believed to be major contributors to error [12] are not as important as previously thought. In particular, the number of phones in a word, the number of minimally short phones in a word (3 frames), and the number of times a triphone was seen in the training data are not good predictors of error. Many other predictors presented here have a much greater impact.

Fortunately, the analysis in this chapter shows that many of these strong predictors appear to be robust across task domain and recognizer. In particular the N-best homogeneity score, language model score, various comparisons with an external reference of best phone sequence, and number of phones in a word all appears to perform identically in both situations. In addition it appears that the behavior of percent match and the phonological distance metric H_{WC} behave similarly in both situations.

Those predictors that appeared to be primarily useful in detecting OOVs in the NAB experiment perform in line with this analysis of the Switchboard results. That is, they give very similar results in the prediction of class *correct*, and do not have a major impact overall in the Switchboard experiment.

Even though the NAB and Switchboard experiments yield very similar results in terms of the cross-entropy reduction figures, the probability-of-correct curves for *correct* and *incorrect* words in the Switchboard experiment actually compare favorably to those in NAB experiment. In the Switchboard results there is a much clearer region in which *incorrect* words are receiving clearly lower $P(C)$ labelings. This is born out by comparison of the related False Alarm/Missed error curve for Switchboard with the analagous NAB curve. In the Switchboard case there is a clear region in the middle in which the sum of False Alarms and Missed Errors is smaller than the original error rate. This is just barely true for the NAB curves. This is the first element of a much more detailed discussion presented. In this discussion the case is made that simply measuring cross-entropy reduction does not capture all the interesting characteristics of a given confidence annotator.

A detailed discussion of learning mechanisms other than decision trees was presented. Generalized linear models, generalized additive models, and neural nets (multi-layer perceptrons) are compared against decision trees under the reduction in cross-entropy measure. Performance in confidence annotation improves if a method that allows each training data point to influence predictions is used. We also see that using methods that capture nonlinearities and interactions between variables improves performance. The neural net and generalized additive model approaches outperform the generalized linear models and decision trees consistently.

The importance of using task-specific performance measures was illustrated through looking at two applications of confidence-annotated Switchboard output. In the first, we filter the recognizer output into high and low quality recognition regions. We discuss how this might be used to identify speakers or conversations for which the recognizer is not performing well. We also discuss the usefulness of confidence annotations done to improve MLLR acoustic adaptation.

Chapter 9

Blame Assignment: When the Truth is Known

This chapter describes an approach to identifying the reasons that recognition errors occur. To do this we combine the computations used to create predictor variables in the confidence annotation work with the error region approach discussed in Chapter4. This combination of "instrumentation" of the recognizer and analysis based on the transcript of what was said allows us to label the causes of errors. Some categorizations of errors can supply training data to automatic corrective training methods that refine acoustic models. Other errors supply language model and lexicon designers with examples that identify potential improvements.

The main goal of the work presented in this chapter is to provide new feedback mechanisms to the system based on the techniques developed in previous chapters. Another more subtle goal is accomplished along the way, however. If we have a reliable error blame assignment technique that can tell us how many errors we could recover by working on a specific system component, then we are no longer tied to the simple word error rate measure when evaluating some particular design change in the system. For instance, if we know that only 15% of our errors on some test were caused by problems with the acoustic models, then we could be satisfied with a change that recovers most of these errors, even though our total reduction in word error rate would not be large. Thus a good blame assignment algorithm can give us the ability to accurately evaluate marginal improvements in system design.

First we will describe the data and computations used in this blame assignment approach. Then we will review the identification and meaning of the types of error regions discussed previously. After the review we introduce a graphic representation of error regions. Following this we describe the blame assignment algorithm in detail together with its results on the NAB data set. Finally we discuss the limitations of the method, describing what we do and do not know about errors after they've been categorized.

9.1 Data Sources Used (ν)

The computations used by the blame assignment algorithm are based on several important sources of information. Some of these come directly from the internal workings of the speech recognizer. Others are additional features computed in parallel. These various sources of information, described in detail in Chapter 2 include:

- Word and phone segmentations and acoustic scores from the best-scoring HYPothesis produced by the recognizer,
- Language model score and source information from the same best-scoring HYPothesis,
- Word and phone segmentations and acoustic scores from the Viterbi alignment of the utterance REFerence transcript,
- Language model score and source information, calculated from applying the recognition language model to the REFerence transcript,
- Characteristics about the words HYPothesized, especially their dictionary pronunciations,
- The results of a parallel "phone-only" decoding, in which the recognizer is not constrained to either phone sequences from the dictionary or word sequences in the language model in its recognition of phone sequences,
- Three distance metrics between basephones, including a simple match count at the frame level, a phonologically-based similarity measure (H_{WC}), and an empirically derived confusion-based distance measure,
- The contents of an N-best list including complete word and phone segmentation and score information for each of its 150 elements.

9.2 Defining Error Regions (ν)

Chapter 4 is devoted to a detailed description of a technique in which regions of error in recognitions are identified and analyzed. The blame assignment algorithm presented in this chapter is an extension of this approach. What follows is a short review of the portions of the method found in Chapter 4 that are used for blame assignment.

An error region is a contiguous set of frames of acoustic data. It starts at the beginning of a word and ends at the end of a (possibly distinct) word. An error region is identified by comparing the REFerence of an utterance against a HYPothesized recognition output. In its strictest configuration, the process that identifies error regions requires that all word segmentations match those in the reference exactly. That is, for each word segment in the reference, the hypothesis must contain a segment with the identical word (including pronunciation variant), start frame, end frame, and thus acoustic score. It is possible that an error region may contain only one word. In the most extreme case, an error region might contain all the words in the utterance.

This definition of *error region* also included two additional factors:

1. A *frame tolerance* can be specified that allows some slack in the comparisons between boundary locations in the reference and hypothesis sequences.

2. The criterion for locating the end of an error region can be changed to reflect the effects of the language model being used by the recognizer. The *window size* parameter can be set to a value of 0, 1, or 2. These values correspond respectively to ignoring language model effects, assuming a bigram language model, and assuming a trigram language model. The parameter is interpreted as the number of successor words that are included in the error region after the hypothesis returns to matching the reference.

Within each identified error region, the following information is collected for later evaluation:

1. *Acoustics:*

    - the total acoustic score of the hypothesized words in the error region,

- the total acoustic score of the reference words in the error region,
- the difference between the two, noting whether the reference or hypothesized acoustic score is better (either HYP/AC or REF/AC).

2. *Language model:*

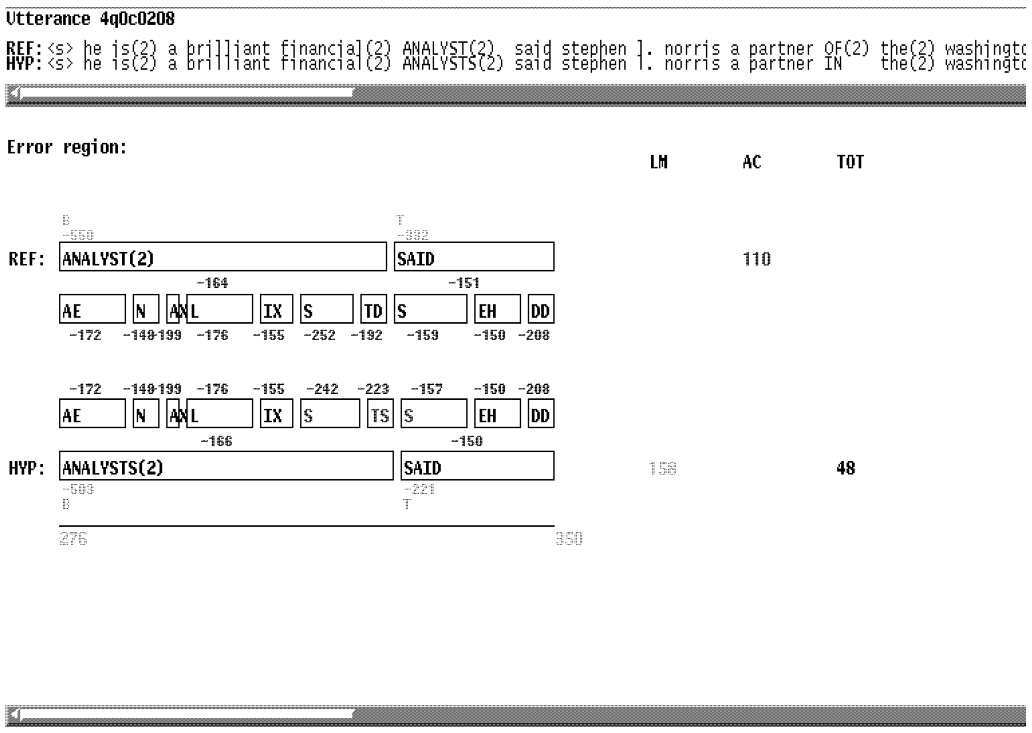
   - the total language model score of all of the words in the hypothesis that fall into the error region,
   - the total language model score of all of the reference words in the error region,
   - the difference between these two, noting which is larger (either HYP/LM or REF/LM).

3. *Totals:*

   - the total combined acoustic and language model score for the hypothesis portion of the error region,
   - the total combined acoustic and language model score for the reference segments in the error region,
   - the difference between these two totals, noting which is larger (either HYP/TOT or REF/TOT).

For any error region three of the values REF/LM, REF/AC, REF/TOT, HYP/LM, HYP/AC, and HYP/TOT will end up containing a score difference. Using these values is possible to classify error regions as falling into one cell of a 2-by-3 matrix, as shown in Table 9.1. The column of the table in which the error region will be placed is determined by whether the REF or HYP had the better overall score (determined by whether REF/TOT or HYP/TOT was chosen). The row is determined by which of acoustic, language, or both acoustic+language, caused the HYPothesis to be chosen (determined by REF/LM vs. HYP/LM and REF/AC vs. HYP/AC).

For an example of how this works, consider the error region displayed in Figure 9.1. In this case the acoustic models preferred the reference sequence. The value of REF/AC is filled in, while the value of HYP/AC is not. The reference portion of the error region scored 110 more acoustic points than did the hypothesis portion. On the other hand, the hypothesized word sequence "ANALYSTS SAID" was preferred by the language model over the reference sequence "ANALYST SAID" by 158 points. The value of HYP/LM is filled



```
Utterance 4q0c0208

REF:<s> he is(2) a brilliant financial(2) ANALYST(2)  said stephen l. norris a partner OF(2) the(2) washingto
HYP:<s> he is(2) a brilliant financial(2) ANALYSTS(2) said stephen l. norris a partner IN    the(2) washingto

Error region:
```

Figure 9.1: An error region caused by the hypothesis language model score overwhelming the otherwise correct reference acoustics.

	REF total better	HYP total better
AC bigger	REF acoustics dominate HYP language model	HYP acoustics dominate REF language model
LM bigger	REF language model dominates HYP acoustics	HYP language model dominates REF acoustics
LM bigger	REF acoustics and language model both better than HYP	HYP acoustics and language model both better than REF

Table 9.1: Error regions can be categorized as belonging to one of these six categories, based on the HYP and REF acoustic and language model scores.

in, while the value of REF/LM is not. This language model preference was strong enough to overcome the otherwise correct acoustics. This error region is categorized in the "HYP language model dominates REF acoustics" cell of Table 9.1. The HYP/TOT value is filled in with the value 48, while the REF/TOT is not filled in. This indicates that the total score of the hypothesis portion of the error region was better by 48 points.

In Figures 9.2 and 9.3 we see a graphic version of the placement of error regions into the cells of Table 9.1. In this scatterplot the x-axis is the acoustic dimension. This axis captures whether HYP/AC or REF/AC was filled in, and what the total value of the one selected was. In the case of our previous example, our error region would take on an x value of -110, as REF/AC is the negative x-axis. The y-axis of the scatterplot is the language model dimension. This axis captures whether HYP/LM or REF/LM was filled in, as well as the total value of the one selected. In our previous example the y value would be +158, as HYP/LM is the positive y-axis. Thus our error region would be put on this plot at the point (-110,158).

The scatter plots in Figures 9.2 and 9.3 there is a dividing line shown (through small squares) along the line $y = -x$. This line divides those errors in the left-hand column of Table 9.1 from this in the right-hand column. All error regions that are placed in the left-hand column of this table have the REF/TOT value filled in, with HYP/TOT empty. This means that the total score for the reference portion of the error region is bigger than the total score for the hypothesis portion of the error region. Any error region in this column will lie below the $y = -x$ line on the scatter plot. Conversely, and region for which the value of HYP/TOT is filled in will lie above this line.

Figure 9.2 represents the error regions found in a typical Switchboard experiment. In this experiment all of the subjects were native speakers of some dialect of North American English. The same is true of the speakers used to train the acoustic models, and the speakers who contributed to the conversations used in the language model training data. By comparison, Figure 9.3 contains error regions generated by non-native speakers of English who appeared in broadcast news shows in a different experiment. Their utterances were recognized by a system trained only on native speech. Notice how the error regions generated by the non-native speakers are almost entirely due to acoustics overwhelming otherwise correct language model scores. This indicates that the language model used was appropriate for use, but the acoustic models were not.

Unfortunately this error region/scatterplot descriptions is not completely satisfying, as it doesn't tell us directly what to correct in our system. We can gain more insight if we look at what caused the errors using the blame assignment algorithm described in the next section.

9.3 Classifying Error Regions (ν)

Figure 9.4 describes a control flow sequence that we pass through in applying the blame assignment algorithm. What follows is a stepwise description of each element of the chart.

First we analyze the utterance as described above to produce a set of error regions. We then proceed with the following tests. As soon as a category for an error region has been found we go on to the next region.

1. *OOV errors*: Is there an out-of-vocabulary (OOV) word in the reference portion of the error region? If so, then we categorize the error region and all of the word errors it contains as belonging to class *OOV*. These errors are not plotted on scatterplots, as it is not possible to score an OOV word in the language model.

2. *Search errors*: Did the reference portion of the error region receive a better score overall than the hypothesis portion? (Is REF/TOT filled in and HYP/TOT is not?) Then if a full search had been used instead of

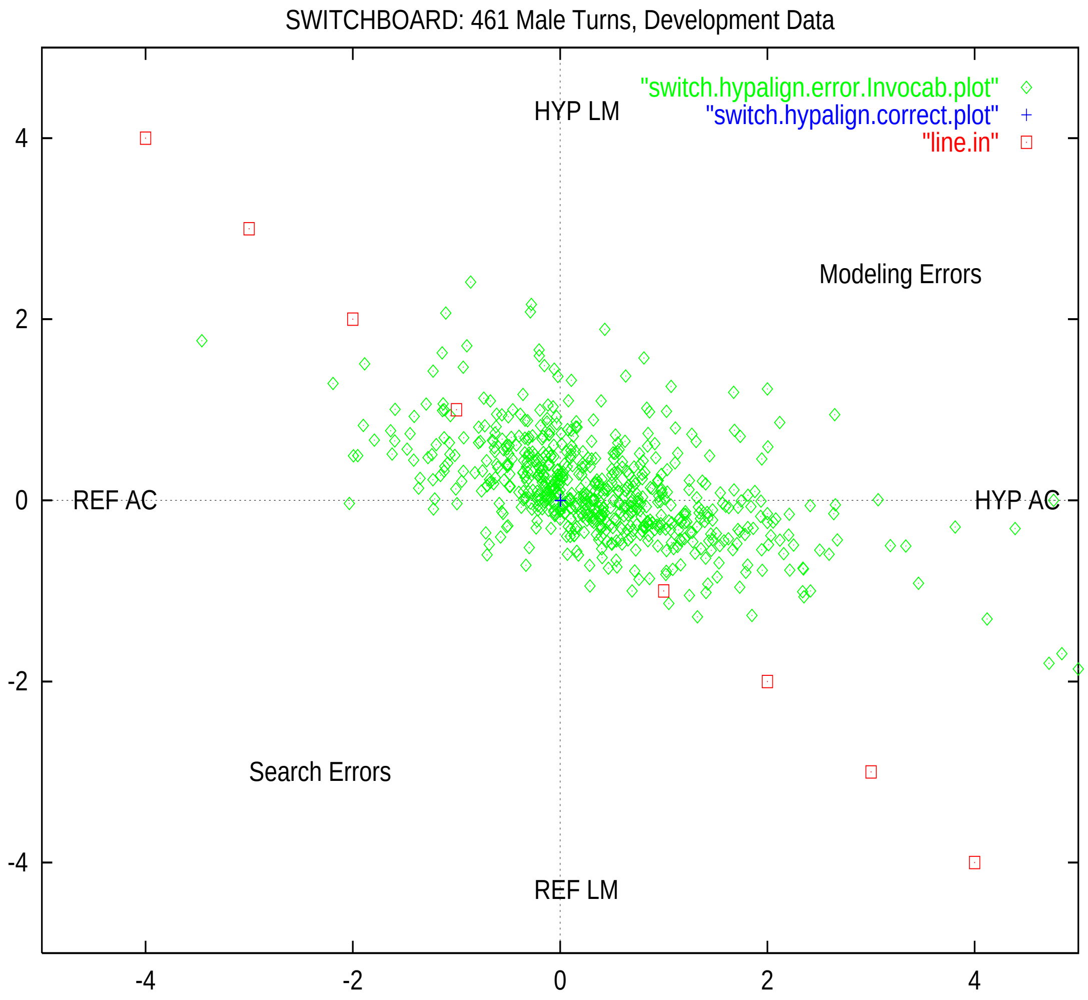


Figure 9.2: An example SWITCHBOARD scatterplot. Placement of the error regions in 461 male utterances from a standard development test set.

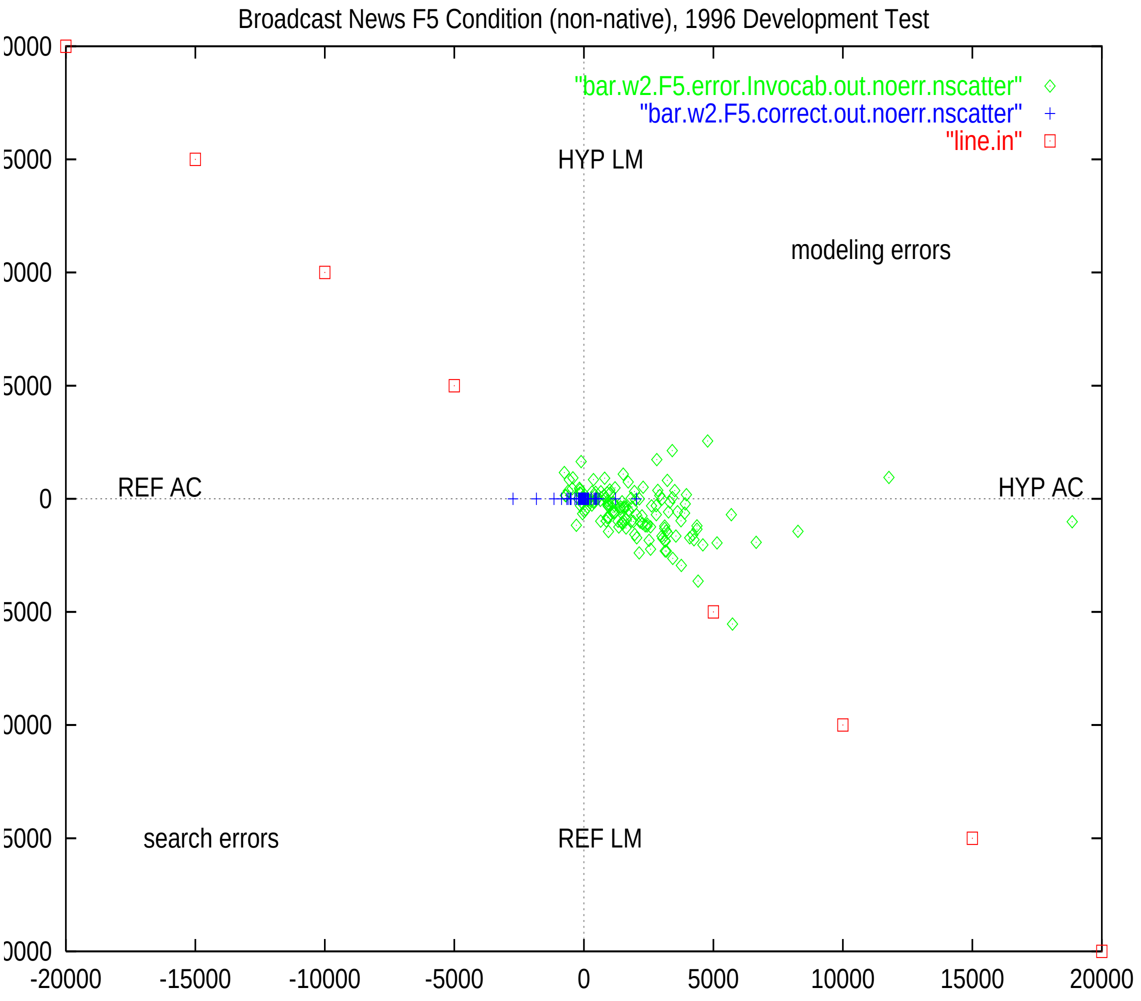


Figure 9.3: A scatterplot of non-native speaker errors from the Broadcast news domain. (F5 portion of the 1995 Development Test.)

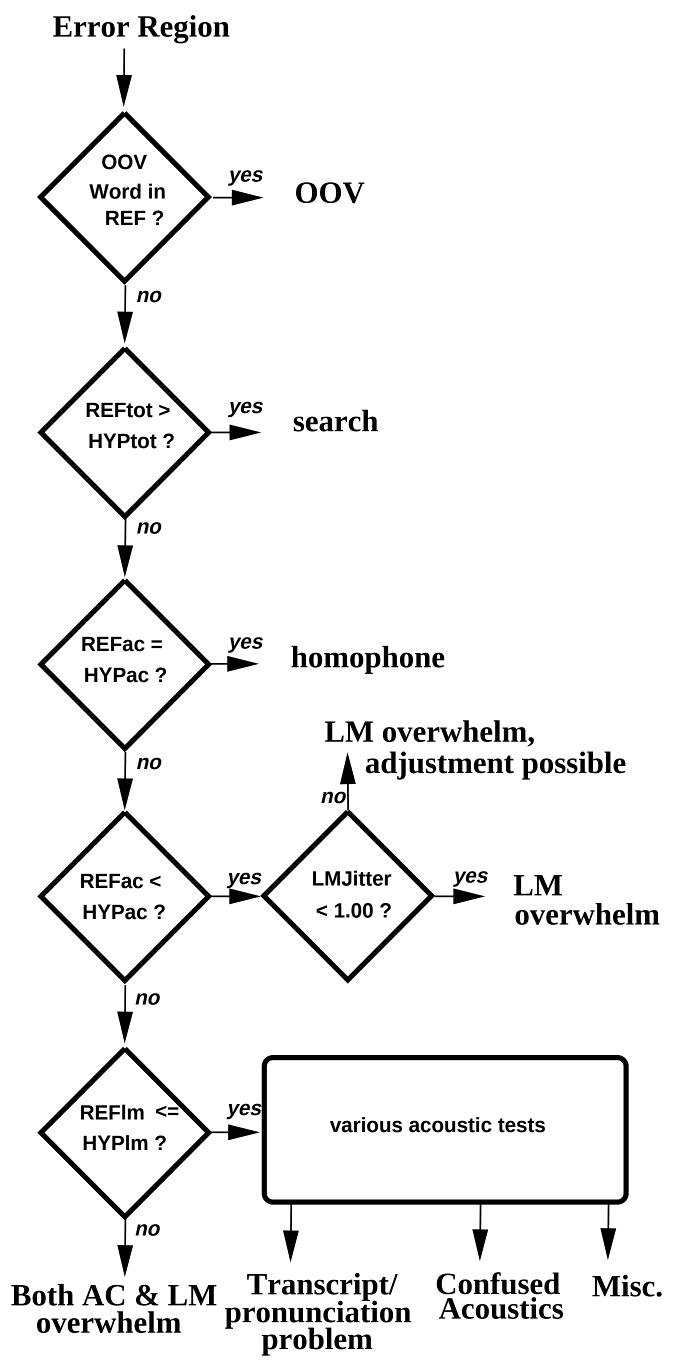


Figure 9.4: The sequence of tests applied by the blame assignment algorithm.

the suboptimal methods we rely on, this answer would have been found. Thus we categorize this error as belonging to class *search*. These errors lie below the $y = -x$ line on scatterplots. All remaining errors lie above this line.

3. *Homophone substitutions*: Did the reference and hypothesis portions of the error region contain the exact same phone sequence and acoustics score, yet the hypothesis was chosen over the reference? (Are REF/AC and REF/HYP both zero?) Then the hypothesis portion of the error region contains a homophone word or word sequence to that found in the reference portion, and the language model chose it incorrectly. Thus we categorize this error as belonging to class *homophone substitution*.[1] These errors lie on the positive y-axis of scatterplots.

4. *Language model overwhelms*: Did the reference acoustics get a better score than the hypothesized acoustics, yet the hypothesis was chosen anyway? (Is REF/AC filled in along with HYP/LM?) Then the acoustic models were capable of making the right decision, but the weighted language model probability was too large to let this happen. In this case we can distinguish between two subcases by asking:

   (a) *Language model overwhelm, LW adjustment possible*: Is the LMJitter value (described in Chapter 8) less than perfect? Is there a possible pair of language weight/insertion penalty (LW/IP) for which the weighted language model probability becomes small enough to let the acoustics make the correct choice? (Is LMJitter less than 1.0?)[2] If so, then we categorize this error as belonging to class *Language model overwhelm: adjustment possible.*

   (b) *Language model overwhelm*: If changing the weights on the language model probability cannot let the correct acoustic decision emerge (if LMJitter = 1.0), then we categorize the error as belonging to class *Language model overwhelm.*

5. *Both acoustic and language model overwhelm*: Did the hypothesis portion of the error region score better both according to the acoustic

[1]This is in fact a special case of the following type of error, "language model overwhelm".

[2]The LMJitter calculation was not available in the NAB experiments.

models and the language model? (We already know that HYP/AC is filled in at this point. Is HYP/LM filled in?) If so, then we categorize the error as being due to *Both acoustic and language model overwhelm*.[3]

6. *Multiple types of acoustic problems*: At this stage we know that the language model score of the hypothesis was not better than that of the reference (REF/LM is filled in). We also know that that the hypothesis acoustic score was better than the reference acoustic score (HYP/AC is filled in). When we reach this point we apply a variety of acoustic tests to try to determine whether:

   - the reference acoustic score is somehow not accurate, or
   - the acoustic models preferred the hypothesis incorrectly.

   Using the tests described next, it is not always possible to tell the difference between these cases. Thus at this point we have a third option, as well: the null categorization "Miscellaneous".

   The acoustics tests proceed as follows:

   (a) *Pronunciation Missing/Transcript Error*: Does the REFerence describe accurately what was really said? Maybe the transcript is wrong, or maybe the pronunciation of the word used is missing from the dictionary. To find out if we're in this class we need to look for gross differences between what we expected to see (based on the phone-only decoding and what's in the hypothesis and N-best list) and what we did see (the HYPothesis). We can also look for evidence that the aligner was straining to fit the REFerence transcript to the acoustic data. To do this we ask:

      i. Does the reference match the phone-only decoding? (If not, then perhaps the reference is not correct.)
      ii. Does the reference match, on average, the phones present in the N-best list? (If not, then this is further evidence that perhaps the reference is not correct.)

[3]This category is difficult to analyze further, but can possibly be "mined" after the fact for examples that meet specific criteria. For example, if we are trying to find a set of examples for corrective retraining of certain basephones, we can look in this category for examples of wrongly scored acoustics of that phone. This does not mean that the error would go away if the acoustic problem were corrected, however.

iii. Are there are one or more minimally short (3 frame) phone durations in the reference? (If so, then perhaps the reference is being forced through phones that are not actually present.)

iv. Are there are extra phones in the phone-only decoding when compared with the reference? (If so, then perhaps the speaker is using an unmodeled pronunciation.)

If two or more of these tests pass, we place the error in the category *Pronunciation Missing/Transcript Error*. If not enough tests pass, we move to the next round of tests.

(b) *Confused Acoustics*: Did some phone model or models give too high a score to the wrong answer? Or perhaps, did the right answer's models not respond with a high enough score? To find out if we are in this case we need to test for evidence of acoustic confusion on the part of the acoustic models. To do this we find the earliest (leftmost) position in the error region in which the hypothesis and reference do not match, and then ask:

i. Is the hypothesis similar to the phone-only decoding under H_{WC}? Under the confusion-based metric?

ii. Is the hypothesis similar to the phones found in the N-best list under H_{WC}? Under the confusion-based metric?

If so, then we categorize this error as belonging to the class *Confused Acoustics*.

If not, then we categorize the error as belonging to class *Miscellaneous*.

Tables 9.2 and 9.3 summarize the computations that are used in the final stage of the blame assignment algorithm described above.

Table 9.4 shows the distribution of categorizations for the NAB development data. The largest single source of errors in this set arises from the presence of out-of-vocabulary words in the utterances. The easiest way to fix these errors is with appropriate vocabulary extensions.

Note that the confidence annotator described in Chapter 6 will predict errors that fall into this first (OOV) category separately from the remaining sets. The goal in the long run is to be able to predict each entry in Table 9.4 according to its class. We currently require knowledge of the reference transcript in order to do this, but a better coverage confidence annotator would remove this requirement.

Distance Between	%Match at Frame Level	H_{WC}	Confusion-based
REF phone vs. Phone-only phone	X		
All phones in N-best list vs. REF phone, avg.	X		

Table 9.2: Tests used to find problems with reference transcripts and the dictionary.

Distance Between	%Match at Frame Level	H_{WC}	Confusion-based
HYP phone vs. REF phone	X		
HYP phone vs. Phone-only phone		X	X
All phones in N-best list vs. HYP phone, avg.		X	X

Table 9.3: Tests used to find regions of acoustic confusion.

All cases of language model overwhelm combined (homophone errors plus the two language model overwhelm cases) create 14.7% of the error regions in the test. These errors should be handed over to those who build language models as examples of how the trigram language model interacted badly with otherwise correct acoustics.

A total of 10.5% of the error regions can be clearly attributed to either incorrect transcripts/pronunciations or confused acoustics. The few potential transcript/pronunciation problems (0.3%) should be handled to a human analyst who decides whether the dictionary should be modified, the transcript should be fixed, or neither. The confused acoustics cases (10.2%) can be used as examples with a corrective training algorithm.

A large group of error regions (36.2%) either cannot be categorized at all (21.5%),[4] or cannot be categorized usefully (14.7%).

[4]For this experiment, this category includes a set of error regions whose reference and hypothesis scores were very close to each other. A discrepancy exists between the acoustic

Category	%Regions
OOV	33.7%
search	9.9%
homophone substitution	2.4%
LM overwhelm	12.3%
Transcript/pronunciation	0.3%
Acoustic Confusion	10.2%
AC+LM Overwhelm	14.7%
Miscellaneous	21.5%

Table 9.4: Categorization of the error regions and errors found in the NAB test data.

9.4 Summary

In this chapter we presented an approach to categorizing the errors that occur in recognition. The approach combines the use of many of the predictor variables used in earlier confidence annotation experiments with the error region analysis technique developed in Chapter 4. The sources of data used to support this technique are the same as were used in the creation of confidence annotators in Chapters 6 and 7.

We discussed how regions of error are identified, and how the acoustic and language model scores used in the recognizer can be compared with similar scores generated for reference transcripts under forced alignment. A general six-way categorization of error regions based on this approach is discussed, together with a graphical representation of the results based on scatter plots of acoustic and language model scores.

A blame assignment algorithm which refines this approach further was

scoring method used in the third pass of the Sphinx-II system (the A* search) and the acoustic scoring method used in the alignment of the N-best lists. On average the total path scores of best hypotheses under these two scoring methods vary within a range of approximately 0.04%. This means that within this 0.04% range the acoustic scores from each measure are indistinguishable. For the experiment reported here, any error region whose reference and hypothesis acoustic scores were within this distance of each other were excluded from the classification process and placed in the "Miscellaneous" category. A total of 11.5% of the error regions examined were treated in this fashion. Thus only 10% of error regions ended up in the Miscellaneous category after undergoing the final phase of acoustic analysis.

then presented. This algorithm places errors into one of the following categories:

- due to out-of-vocabulary (OOV) word spoken,
- search error,
- homophone substitution,
- language model overwhelming correct acoustics,
- transcript/pronunciation problems,
- confused acoustic models, or
- miscellaneous/not possible to categorize.

The possible uses of the various categorized errors were discussed. This included both the potential use of categorized errors in automated training algorithms and the possibility of further human analysis for appropriate adjustments to language models, transcripts, and dictionaries. Figures for the distribution of error types in the NAB development test set were presented.

Chapter 10

Lessons Learned

In this chapter we begin with a summing up of what we have learned about how it is that our recognizers behave when they make errors. After this we describe a list of system design elements that are important for the support of error analysis, blame assignment, and confidence annotation work. We then summarize our confidence annotation and blame assignment results. Finally, we describe several ongoing efforts in a the section "Future Work".

10.1 Error-Related System Behaviors

This section is a summary of the observed behaviors of the Sphinx-II system that are clearly related to the production of errors. (Cases in which the experiments using Janus also indicate similar behaviors are marked.) These observed behaviors are presented with an eye toward possible leverage points for improvements in system design. They are presented in order of apparent importance as determined by their incremental contributions to the confidence annotation results of Chapters 6 and 7.

1. Most errors occur in regions in which the recognizer experiences *confusion between multiple possible decodings.* This can be shown using a variety of measures. The most powerful measure is called "N-best homogeneity". For a particular hypothesized word this measure tells us what portion of the elements of the N-best list pass through the same word at the same location, weighted by total path score. Doing this calculation is at the phone level is even more powerful than at the word level. Just doing this calculation alone at the phone level (on the

NAB data) helps us fine almost 30% of the incorrect phones without suffering from a high false alarm rate. Another good measure of this phenomenon is the count of how many distinct words, on average, were present in the N-best list at the location of the hypothesized word.^{Janus}

2. A parallel phone-only decoding can be used as an external reference point to find regions in which the results of the search constrained by language model and dictionary sequences produces a phone sequence dissimilar from search not subject to these constraints. When used in conjunction with the previous measures, this measure can pick out regions in which the recognizer correctly resolves potential confusion between multiple possible decodings. Because the phone-only decoding is only 72-73% accurate, we rely on this measure mostly to *corroborate that the correct decision was made despite evidence of potential confusion.*

3. Another independent source of constraint-free sequences of acoustic matches comes directly from the acoustic scoring calculations done at every frame. From this stream we can identify the best-scoring basephone at any frame. This sequences is much noisier than the phone-only decoding, as it is not constrained to pass through entire phone state sequences. Especially when smoothed mildly^{Janus}, this measure is also useful in *corroborating that the correct decision was made despite evidence of potential confusion.* Two distance metrics are of use in comparing the phone-only decoding with the recognition hypothesis. The first is a simple frame-level percent match. The second, slightly more powerful, is a measure of phonological similarity referred to in this document as H_{WC}.

4. The identification of the best basephone at every frame is based on the calculation of the best possible acoustic score at each frame. Thus for any particular set of frames in the utterance we can calculate what the best possible acoustic score would be under no phone, dictionary, or language model constraints. For any given hypothesized word the distance between this optimal score and the actual acoustic score is an indicator of how likely the word is to be correct. Very close matches indicate that the presence of phone, dictionary, and language model constraints did not prevent the recognizer from picking a close to optimal acoustic decoding. Very distant matches indicate that the recog-

nizer was possibly forced to *trade off a good acoustic match against a passable overall path score*. This predictor is improved noticeably when measured at the phone level instead of the word level. This predictor is used in conjunction with those above. When there is evidence of potential confusion from the N-best list, and the phone-only decoding cannot confirm that the hypothesized word is correct, this measure is used to confirm that an error is likely.^{Janus}

5. In addition to knowing the best basephone at any frame, in a semi-continuous system such as Sphinx-II we can know the overall rank of each basephone at every frame. If we compare the hypothesized phones with this rank, we can get a sense for "how far down the list" we had to go in order to come up with our hypothesis. This measure is used in the same fashion as the previous. *When other measures fail to tell us whether potential confusions have been overcome, this can help identify error-prone situations.*

6. Overall, if a word receives a *poor language model score*, it is more likely to be incorrect.^{Janus} This effect is small for most errors. However, for the cases in which the recognizer is trying to "cover" *out-of-vocabulary words* (OOVs), this effect becomes more pronounced. In cases in which there is evidence of acoustic confusion in the N-best list and the language model score is also very low, the probability that an OOV is present is the highest. This is a reflection of the manner in which the recognizer deals with OOVs. Because in general the acoustic score plays a stronger role in overall scoring than does the language model, the recognizer's first goal is to find a reasonably close acoustic match from among the dictionary pronunciations available. The second goal is to find a sequence of these pronunciations which is acceptable to, but not necessarily excellent, under the language model. For this reason we typically see close but imperfect acoustic matches and nonsensical and low-scoring language model sequences in the vicinity of OOV words. The *source of the language model score* is also a predictor in this case, as backing off to low-order N-grams in the language model occurs far more frequently in the vicinity of OOVs than elsewhere.

7. Another good indicator of the presence of OOV word in the input is the count of how many times the hypothesized word was seen in the acoustic training data. This measure is applied in cases where there

is evidence of extreme acoustic confusion in the N-best list. In such cases, if the hypothesized word was *never seen in the acoustic training data*, there is a high probability that an OOV is present in the input.

8. In read speech applications, such as the NAB, which contain a wide variety of lexical items, the average length of dictionary pronunciations is longer. When recognizing in such domains, the *longer words in the dictionary are more likely to be recognized correctly than the shorter items.* This is related to the very first point in this list. The longer a word is, the more likely it is that all other words will have been pruned from the Viterbi search's beam by the time the end of the word is reached. This means that in the N-best list there will be fewer possibilities for confusion.

9. In read speech applications, such as the NAB, which contain a very large vocabulary, it is quite common to have triphones (both within-word and cross-word) modeled in the system which were never seen in the acoustic training data. It is still the case that *unseen triphones are more likely to be hypothesized incorrectly* than are seen triphones.

In the Switchboard/Janus experiments, the design of the Janus system allowed us to easily see the "push in/pop out" effects created by varying the language weight (LW) and word insertion penalty (IP) across a large grid. If, for any LW/IP pair, a hypothesized word fell out of consideration, then the likelihood that it was incorrect was high. This method is not as good at identifying correct words – many words that never "pop out" are actually incorrect. But used in conjunction with other measures from the above list, this was the most powerful base predictor in the Switchboard experiments.

10.2 System Design Considerations

During the course of the development of this thesis, various error analysis, blame assignment and confidence annotation experiments were tried on a variety of recognizers and databases. In total four recognition systems were used: Sphinx-II, Sphinx-III [50], Janus, and the commercial version of HTK. Not all of these experiments are reported in this document. This is, in part, due to problems experienced in trying to get the various recognizers and their forced alignment counterparts to work smoothly in producing the type of input necessary for the analyses presented here.

This section is a summary of the engineering-level design issues that influence how easy or difficult it is to adapt a recognizer for use in the error analysis, blame assignment, and confidence annotation approaches described in this work. The goal in describing these is to encourage system designers to consider these issues.

- The space optimizations that we make in the recognizer to support our expensive acoustic model computations make it impossible to get an accurate picture of the word-internal phone segmentations and scores that the recognizer relies on in its decision making. In order to overcome this is was necessary to use forced alignment to create *ex post facto* an image of the state and phones sequences the recognizer probably traversed in making its hypotheses at the word level. While it may not be possible to dump exact segmentation and score information from the initial Viterbi passes, anything that can be done to make a *post hoc* alignment as similar to the initial computation as possible is very helpful. For example, the Janus system provides functionality that supports alignment constrained within word boundaries for its lattice word elements.

- Surprisingly, many recognizers use separate code to implement their recognition search and the forced alignment of acoustic data against transcripts. (In most cases this is because forced alignment is typically only used during acoustic training.) The problem with this is that often this results in slight differences between the assumptions made about possible state sequences traversable by the recognizer vs. the aligner. For example, it is quite common that a recognizer will allow optional silence and noise word insertions between words under different rules than the those constraining the aligner. For error analysis and blame assignment work it is very important that the aligner and the recognizer share the exact same possible state transitions, otherwise detailed comparison is impossible in many cases. This is not at odds with the theoretical underpinnings of speech recognition. In fact, the forced alignments should represent the exact Viterbi (or other) alignment approach present in the recognition passes, but simply constrained to a small subset of all possible paths.

- The ability to save compact lattice representations for repeated use is important. The ability to do quick lattice scoring is also very useful, especially under varying language weight and insertion penalty conditions. These lattices would ideally contain word and phone-level acoustic scores and segmentation information. Word language model scores are also necessary. Ideally some sort of tag describing how the language model score was calculated within the backoff scheme employed would also be useful.

- It is very useful to have a phone-only recognizer that employs the same or similar acoustic models as the main recognizer.

- It is very useful to be able to calculate the distribution of acoustic scores and their corresponding basephones at each frame.

10.3 Confidence Annotation

In this section we summarize the contributions made to confidence annotation research in this thesis. The main contributions of this thesis in confidence annotation are:

- Confidence annotation systems on two separate tasks and systems that perform substantially better than baseline and previously reported systems.

  The best confidence annotation system at the word level in the NAB read speech task reported here yields a 20.9% overall reduction in cross-entropy. The base rate of error for this task is 16%.

  In the same experiment repeated at the phone level of abstraction, the base rate of error is 6% for female speakers and 7% for males speakers. Even though the base rate of the error phenomena is 2-3 times lower, our ability to reduce the cross-entropy is much greater, at 29%.

  Combining these two approaches and moving to the new task domain of Switchboard telephone conversations, we are able to achieve a 21.3% reduction in cross-entropy. Moving from decision trees to generalized additive models as the learning mechanism, this number improves to 24%.

- A detailed analysis of a wide variety of predictor variables, together with a discussion of how general the behavior of these predictor variables is across two systems and three annotation tasks.

  Section 10.1 summarizes the main predictor variables used in the confidence annotation experiments and their key interactions, as well as their robustness across recognition task and system. The summaries of each of Chapters 6, 7, and 8 give more detail about individual experiments. A detailed comparison of predictor performance across domains and systems appears in Sections 8.1 and 8.2.

- A discussion of the power of doing phone-level annotation to produce an acoustic confidence measure, and the propagation of the results up to the word level.

  Details of the improvements in acoustic predictor behaviors at the phone level are covered in Sections 7.2 and 7.3.

- A detailed comparison of the performance of the decision tree learning mechanism against three alternate approaches: generalized linear models, generalized additive models, and neural networks.

  Details are presented in Section 8.3.3.

- A discussion of how doing multi-class confidence annotation can provide both a "poor man's" OOV detector and improved overall confidence annotation.

  Details are presented in Section 6.10.

- A discussion of the value of using the cross-entropy reduction metric in conjunction with $P(Correct)$ curves and their commensurate False Alarm/Missed Error tradeoff curves.

  All three Chapters 6, 7, and 8 contain details on the importance of looking at $P(Correct)$ curves and False Alarm/Missed Error tradeoffs when evaluating confidence annotators in conjunction with the cross-entropy reduction measure.

- A discussion of the necessity of doing application-specific evaluation of confidence annotation.

Sections 6.9.2 and 8.3.3 describe four different (potential and actual) uses of confidence annotations, together with arguments in favor of application-specific evaluation of all confidence annotators.

10.4 Blame Assignment

The recognizer "instrumentation" developed for the confidence annotation work can be combined gracefully with the error analysis approach developed in Chapter 4 to do blame assignment for a majority of recognition errors.

This blame assignment approach was presented in Chapter 9. We discussed how regions of error are identified, and how the acoustic and language model scores used in the recognizer can be compared with similar scores generated for reference transcripts under forced alignment. A general six-way categorization of error regions based on this approach is discussed, together with a graphical representation of the results based on scatter plots of acoustic and language model scores.

A blame assignment algorithm which refines this approach further was then presented. This algorithm places errors into one of the following categories:

- due to out-of-vocabulary (OOV) word spoken,
- search error,
- homophone substitution,
- language model overwhelming correct acoustics,
- transcript/pronunciation problems,
- confused acoustic models, or
- miscellaneous/not possible to categorize.

A visual error analysis tool based on this approach is in use at a variety of sites in the speech recognition community. (See Appendix B.)

The possible uses of the various categorized errors were discussed. This included both the potential use of categorized errors in automated training algorithms and the possibility of further human analysis for appropriate adjustments to language models, transcripts, and dictionaries. Figures for the distribution of error types in the NAB development test set were presented.

10.5 Future Work

This thesis work leads in many directions. Improvements to the confidence annotators themselves are possible. A wide variety of applications of confidence annotators are possible. Error analysis and blame assignment can be used together to find points of improvement in our systems. What follows is a list of future projects, some of which are already in progress.

- *Improving the annotators*:

  Phone-specific acoustic predictors: It is very likely that specific phones or classes of phones will yield increased error-predictive power under several of the acoustic predictors if they are broken out and treated separately. Seeking these special response relationships and incorporating them into our predictors may improve confidence annotation performance. Some promising results along these lines are reported in [52].

  Topic-dependent words: Work in topic-dependent language modeling [55] indicates that error distributions among words considered to be topic-dependent are different than among those considered to be topic-independent. Incorporating these differences into our models may improve confidence annotation performance, especially in topic-driven applications.

- *Application of confidence annotators*:

  Tagging multi-media databases: In collaboration with a multi-media database project, we are currently investigating the application of confidence annotation to improving document retrieval. [21] [22].

  Finding the optimal LW/IP pair for a speaker: Initial analysis indicates that simply trying to optimize the LW/IP pair for a single speaker within the Switchboard task could improve base word error rate substantially [15]. This could potentially be accomplished via repeated rescoring of lattices under a variety of LW/IP pairs and selecting the values that yield the best confidence rating over a complete conversation side.

  Dynamic adaptation of LW: Initial results from IBM [41] using a rudimentary confidence annotator indicate that dynamic adaptation of the contributions of the acoustic and language models may be possible.

Focusing language models on error-prone regions: Syntax- and semantic-based language models have not yet been able to outperform the simple trigram language model in speech recognizers. This is in part due to a high rate of fixing "errors" in recognizer output that weren't actually wrong. Using confidence annotators in conjunction with these language model approaches, we may be able to focus the more sophisticated language models on regions which are highly error-prone, thereby reducing the problem of fixing things that aren't broken.

- *Using the blame assignment algorithm*:

  Detecting dictionary problems: Similar work involving the use of phone-only decodings and reference forced alignments [38] to detect dictionary pronunciation problems is under way. Initial results indicate that the slightly more sophisticated approach to detecting transcript/dictionary problems in the blame assignment algorithm is reliable.

  Identifying training examples for corrective acoustic training: Similar work involving the use of the best path in a search lattice as a reference point to identify acoustic errors that should be corrected for decreased confusability is in progress [60].

  Examples for language modelers: A small but possibly important number of errors categorized by the blame assignment algorithm will be shown to language modeling experts for analysis.

Appendix A

Phone Frame Confusions

The phone confusion table at the frame level for the Sphinx-II/NAB experiments:

REF\HYP	AA	AE	AH	AO	AW	AX	AXR	AY	B	BD	CH	D	DD	DH	DX	EH	ER	EY	F	G	GD	HH
AA	7644	108	111	160	41	20	2	117	2	0	0	0	0	4	1	14	0	0	0	0	0	1
AE	8	13838	160	8	96	155	0	43	5	1	0	1	0	19	0	94	0	152	0	0	0	14
AH	15	25	8268	89	25	23	0	57	3	0	0	0	11	2	0	65	0	0	0	0	2	4
AO	68	37	42	10329	12	11	11	11	0	0	1	1	2	0	0	0	11	0	19	0	0	4
AW	0	41	34	44	3347	12	6	0	1	0	0	1	0	0	1	0	0	0	0	0	0	2
AX	27	201	62	58	3	23567	95	5	9	4	4	5	81	43	3	74	3	36	8	2	0	20
AXR	3	11	9	9	3	77	12477	9	10	0	1	0	7	4	7	28	99	4	2	0	1	1
AY	54	46	71	8	5	38	4	11405	1	0	0	0	3	0	0	20	0	13	0	0	0	12
B	1	0	13	2	1	10	7	1	6250	0	0	18	34	46	7	1	0	5	0	3	0	0
BD	0	1	2	0	0	0	0	1	4	88	0	0	0	0	0	0	0	0	0	0	12	0
CH	3	1	0	0	0	0	3	1	0	0	3178	0	4	0	0	0	0	3	0	0	0	0
D	3	1	0	4	3	5	5	0	18	0	0	5581	83	38	7	11	0	3	10	7	0	5
DD	1	0	0	0	2	41	9	2	16	0	4	23	5908	62	7	7	3	11	11	2	0	3
DH	0	8	1	1	1	58	2	0	22	0	0	0	39	5930	0	5	0	0	9	0	0	9
DX	0	0	1	0	0	8	5	9	5	3	0	8	7	11	3091	2	2	4	0	0	0	0
EH	0	386	98	0	20	83	42	15	0	0	1	2	12	3	3	16366	9	40	1	2	0	17
ER	5	0	0	0	0	8	101	0	0	0	0	1	0	1	0	23	4120	10	0	0	0	1
EY	2	28	0	0	0	13	7	3	5	0	0	1	2	15	4	15	0	13177	0	0	0	0
F	0	1	0	4	0	2	1	0	7	0	0	0	14	5	0	1	4	0	12125	0	0	11
G	1	2	2	0	0	3	2	0	0	0	0	12	13	0	10	5	0	5	0	3050	0	0
GD	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	89	0
HH	1	5	4	2	2	6	4	1	0	0	4	0	3	0	0	5	1	1	3	10	0	4860
IH	0	87	4	0	5	46	5	18	5	0	0	0	11	0	0	102	3	52	1	5	0	0
IX	0	24	19	0	0	223	29	16	3	0	0	5	9	15	3	57	1	62	1	3	0	13
IY	8	18	4	0	0	53	8	0	4	0	0	1	11	3	2	14	0	183	1	3	0	8
JH	1	1	0	0	0	6	3	0	6	0	10	4	19	8	7	0	0	5	1	6	0	0
K	1	1	4	10	0	5	4	0	0	0	3	0	7	5	5	2	0	0	0	60	0	33
KD	0	1	0	0	0	10	0	0	0	0	0	1	7	12	0	0	0	1	0	0	0	0
L	10	6	36	48	31	44	10	17	18	0	0	6	21	13	0	7	1	7	11	2	0	6
M	0	1	4	0	1	15	3	1	16	0	3	5	26	23	7	4	0	1	0	0	0	0
N	18	44	26	10	7	96	13	9	0	9	0	21	204	12	8	28	1	22	4	0	0	20
NG	0	2	0	0	0	4	1	0	2	0	0	2	3	12	6	0	0	33	0	0	0	3
OW	1	14	62	76	66	55	0	0	9	0	0	0	2	1	0	71	5	0	0	0	0	3
OY	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	3	0	0	0	0
P	25	27	0	8	0	13	3	1	48	4	0	0	21	34	0	3	4	1	20	9	0	0
PD	0	0	2	0	0	2	3	0	0	0	0	3	0	5	0	1	0	0	5	0	0	0
R	39	20	22	58	0	33	219	17	19	0	0	3	13	8	0	23	49	39	12	6	0	2
S	0	1	2	4	0	21	2	0	0	0	0	5	9	40	0	7	0	5	19	2	0	3
SH	0	0	0	0	0	2	2	0	0	0	19	0	1	0	0	0	0	0	0	0	0	0
SIL	11	13	13	10	0	71	3	0	34	0	20	7	164	51	0	1	0	2	36	1	0	100
T	2	1	3	1	4	18	1	0	4	0	0	43	34	13	4	8	2	1	7	8	0	8
TD	0	12	19	0	7	47	3	2	14	8	2	15	78	23	3	8	0	10	0	2	0	12
TH	0	1	0	0	0	8	0	0	0	0	0	5	0	50	0	0	0	0	39	0	0	0
TS	0	0	0	0	0	7	0	0	0	0	1	8	4	2	0	0	0	0	5	0	0	3

UH	8	4	12	31	0	38	4	16	0	0	0	0	0	0	0	8	14	0	5	0	0	1
UW	0	4	0	7	0	37	9	0	1	0	0	0	3	0	0	3	4	17	0	0	0	0
V	0	2	6	0	0	35	2	3	28	0	0	4	19	66	9	3	1	1	51	0	0	0
W	6	0	9	15	0	20	7	2	5	0	0	0	1	8	0	3	0	0	21	0	0	3
Y	0	0	0	0	0	2	0	3	0	0	0	0	2	1	0	0	0	0	0	3	0	12
Z	0	2	1	1	0	14	14	1	2	0	0	13	20	15	6	7	0	1	34	0	0	0
ZH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SILb	0	0	0	0	0	0	0	0	3	0	0	0	0	5	0	0	0	0	0	0	0	0
SILe	0	0	0	0	0	4	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0

REF\HYP	IH	IX	IY	JH	K	KD	L	M	N	NG	OW	OY	P	PD	R	S	SH	SIL	T	TD	TH	TS
AA	10	10	0	1	2	0	25	3	6	0	32	5	0	0	25	0	0	1	0	0	0	0
AE	46	45	38	0	8	2	8	4	33	1	0	0	11	0	10	0	1	6	4	12	0	0
AH	0	10	0	0	1	0	4	9	12	7	41	5	0	1	2	2	0	3	0	5	0	0
AO	0	8	0	0	2	3	22	3	4	0	35	9	2	0	9	0	0	5	1	0	0	0
AW	0	0	0	0	0	0	9	1	0	0	4	0	0	0	0	0	1	3	3	0	0	0
AX	59	237	71	3	14	5	71	34	138	10	81	3	4	8	53	17	10	18	23	29	0	10
AXR	11	51	20	3	2	0	2	6	16	0	7	0	0	0	202	0	2	6	4	11	0	0
AY	6	13	0	0	0	0	17	0	17	1	9	0	0	0	10	1	0	11	6	7	0	0
B	1	2	2	0	0	0	7	42	17	2	4	0	23	2	8	1	0	19	0	4	0	0
BD	0	0	0	0	0	5	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
CH	0	1	0	51	0	0	0	0	4	0	0	0	0	0	0	1	44	2	30	3	0	9
D	5	7	6	4	4	2	5	9	89	0	0	0	6	0	2	7	0	9	82	30	0	13
DD	2	27	21	16	8	4	12	35	181	9	0	0	8	0	14	10	5	25	27	62	7	0
DH	9	8	0	0	4	0	12	9	32	19	0	0	0	0	0	20	0	21	9	29	13	0
DX	1	7	17	0	3	6	0	3	25	0	1	0	0	4	1	3	8	0	33	20	0	0
EH	113	81	13	2	0	1	10	2	40	1	10	0	1	0	14	5	0	8	1	7	0	2
ER	10	3	1	0	2	0	2	1	2	0	0	0	1	0	41	0	0	0	2	0	0	0
EY	32	31	187	0	0	5	7	2	16	3	0	0	0	0	1	1	0	7	2	17	0	0
F	1	2	0	0	0	18	7	0	2	1	0	0	23	0	4	80	4	37	1	19	20	11
G	5	4	17	0	50	22	10	13	43	14	6	0	32	0	8	12	0	3	12	16	0	0
GD	0	0	0	0	0	20	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HH	5	19	8	5	27	1	16	0	12	3	3	0	0	0	1	3	0	68	15	10	0	0
IH	8906	45	48	0	0	1	7	3	13	8	0	3	3	2	9	4	0	12	4	3	4	0
IX	103	14318	97	2	3	2	8	6	67	9	9	1	0	0	28	7	2	4	4	31	1	2
IY	38	57	18821	0	4	2	9	1	13	18	0	0	6	4	4	3	0	18	7	12	0	0
JH	0	7	2	3795	0	7	0	0	8	0	0	0	0	0	4	6	9	6	55	1	0	26
K	0	13	7	1	14455	51	3	2	2	0	1	0	33	4	2	4	0	38	45	42	0	1
KD	2	1	13	0	46	3213	1	0	10	1	1	0	10	5	1	3	0	42	2	69	1	0
L	9	10	15	0	1	1	18421	37	16	14	138	10	4	0	25	6	0	2	0	9	1	2
M	9	4	2	0	7	3	27	12741	193	7	4	0	0	0	4	3	0	22	0	2	0	0
N	33	55	20	8	6	15	20	219	28567	125	11	0	0	0	18	18	2	16	6	74	3	7
NG	6	12	25	0	1	1	4	16	94	4721	0	0	0	0	0	1	0	1	1	0	0	0
OW	73	11	7	0	0	0	122	0	5	10	7698	5	0	3	11	4	0	9	0	9	0	6
OY	0	6	10	0	0	0	0	0	1	0	21	1522	0	0	4	0	0	0	0	0	0	0
P	7	3	3	0	67	0	1	11	4	0	1	1	11627	18	4	0	0	81	27	14	11	4
PD	0	3	0	0	0	29	2	0	4	0	4	0	44	1021	17	0	0	17	3	30	6	0
R	10	28	39	2	2	4	27	28	6	1	7	4	4	0	15870	1	1	17	16	10	1	0
S	2	4	0	4	1	9	2	1	2	0	4	0	4	0	0	31833	8	9	34	19	25	96
SH	0	0	1	0	5	0	0	0	0	0	0	0	0	0	0	71	6916	7	17	0	0	0
SIL	8	14	2	0	13	20	19	5	24	0	15	0	49	0	12	8	0	54831	26	88	6	3
T	0	0	15	16	29	7	7	14	19	4	0	0	0	16	9	27	15	26	15379	97	15	29
TD	5	30	10	2	33	63	9	21	100	0	1	3	18	0	7	109	3	62	74	7944	0	54
TH	2	0	0	0	1	4	1	0	2	0	0	0	14	0	2	10	0	10	35	10	2169	0
TS	0	2	1	0	19	0	0	0	0	5	0	0	14	3	0	126	1	13	5	84	9	3738
UH	0	0	0	0	4	0	21	0	3	0	7	0	0	0	3	0	0	0	1	2	0	0
UW	23	9	40	0	3	2	122	2	19	6	45	0	0	1	19	1	0	0	1	10	0	0
V	1	9	6	2	4	0	5	13	56	0	2	0	13	0	19	6	0	18	4	32	2	0
W	3	0	1	0	13	0	46	22	2	0	20	11	4	0	10	0	0	41	0	7	0	1
Y	18	0	60	0	1	3	0	2	1	0	2	3	1	0	3	1	0	5	12	0	0	0
Z	2	34	11	0	1	0	1	7	30	3	1	0	0	0	12	390	13	27	23	3	7	57
ZH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SILb	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1460	0	0	0	0
SILe	0	0	9	0	0	3	0	0	13	0	0	0	0	0	0	1	0	415	0	1	0	1

REF\HYP	UH	UW	V	W	Y	Z	ZH	SILb	SILe
AA	0	0	0	12	0	0	0	0	0
AE	1	0	7	1	4	4	0	0	0
AH	0	0	1	12	0	1	0	0	0
AO	0	0	5	65	0	1	0	0	0
AW	0	0	0	0	0	0	0	0	0
AX	8	52	21	12	1	26	0	0	3
AXR	5	0	5	2	0	6	0	0	0
AY	0	0	5	4	0	1	0	0	1
B	0	1	28	8	0	2	0	0	0

BD	0	0	0	0	0	0	0	0	0
CH	0	2	0	0	0	0	0	0	0
D	0	1	8	3	0	33	0	0	0
DD	0	0	42	18	1	35	0	0	0
DH	0	2	13	0	0	17	0	0	0
DX	0	3	0	0	0	0	0	0	0
EH	0	0	6	3	18	2	0	0	0
ER	0	0	3	0	0	1	0	0	0
EY	0	0	3	0	0	1	0	0	0
F	0	0	29	1	0	14	0	0	0
G	0	0	0	10	16	0	0	0	0
GD	0	0	0	0	0	0	0	0	0
HH	2	2	2	36	13	0	0	0	0
IH	18	9	4	1	22	4	0	0	0
IX	0	27	6	5	10	41	0	0	0
IY	3	16	0	1	22	9	0	0	0
JH	0	1	0	1	0	5	19	0	0
K	0	0	3	11	0	2	0	0	0
KD	0	0	8	0	0	0	0	0	0
L	5	14	17	100	0	5	4	0	0
M	0	10	18	10	5	4	0	0	0
N	0	16	24	15	24	34	0	0	3
NG	0	1	0	10	3	0	0	0	0
OW	0	24	6	4	0	11	0	0	0
OY	0	8	0	3	0	0	0	0	0
P	0	0	33	10	4	1	0	0	0
PD	0	0	0	0	0	0	0	0	1
R	8	13	7	25	0	2	0	0	7
S	0	0	12	5	2	553	0	0	0
SH	0	0	6	0	3	16	0	0	1
SIL	0	0	16	23	0	32	10	1	15
T	0	3	4	0	0	0	0	0	0
TD	0	0	29	10	0	25	0	0	0
TH	0	0	2	0	0	15	0	0	0
TS	0	0	4	0	0	89	0	0	0
UH	824	1	2	11	0	0	0	0	0
UW	14	5851	3	12	31	7	0	0	0
V	0	9	7265	11	3	30	0	0	2
W	3	2	0	7587	0	2	0	0	0
Y	1	8	0	0	4417	0	0	0	0
Z	0	5	24	2	2	19619	0	0	0
ZH	0	0	0	0	0	0	435	0	0
SILb	0	0	0	0	0	0	0	32119	0
SILe	0	0	0	0	0	11	0	0	49850

Appendix B

The Visual Error Analysis Program (visERA)

The visual error region analysis programs described in detail in Chapter 4 are available via ftp and tape distributions. Installation requires the *gcc* compiler and a fully installed version of *Tcl/Tk*.

Contact the author at *chase@cs.cmu.edu* for more information.

Bibliography

[1] F. Alleva, X. Huang, and M. Hwang. An Improved Search Algorithm for Continuous Speech Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1993.

[2] A. O. Asadi. *Automatic Detection and Modeling of New Words in a Large-Vocabulary Continuous Speech Recognition System.* PhD thesis, Electrical and Computer Engineering, Northeastern University, August 1991.

[3] J. Baker and etc.. Large Vocabulary Recognition of Wall Street Journal Sentences at Dragon Systems. In *DARPA Speech and Language Workshop*, February 1992.

[4] L. E. Baum. An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. *Inequalities*, 3:1–8, 1972.

[5] F. Beaufays and M. Weintraub. Neural-network Based Measures of Confidence. LVCSR Hub5 Workshop Presentation, October 1996.

[6] William Byrne. Personal Communication. unpublished, 1996.

[7] J.M. Chambers and T.J. Hastie. *Statistical Models in S.* Wadsworth and Brooks, Pacific Grove, CA, 1992.

[8] L. Chase and R. Rosenfeld. Error-Responsive Feedback to Our Speech Recognizers. LVCSR Hub5 Workshop Presentation, October 1996.

[9] J. Cohen. Personal Communication. unpublished, 1996.

[10] Stephen Cox and Richard Rose. Confidence Measures for the Switchboard Database. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 511–514, May 1996.

[11] S.B. Davis and P. Mermelstein. Comparison of Parametric Representations of Monosyllabic Word Recognition in Continuously Spoken Sentences,. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(4):357–366, August 1980.

[12] Ellen Eide, Herbert Gish, Philippe Jeanrenaud, and Angela Mielke. Understanding and Improving Speech Recognition performance Through the Use of Diagnostic Tools. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 221–224, 1995.

[13] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal. The Karlsruhe-Verbmobil Speech Recognition Engine. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[14] M. Finke and T. Zeppenfeld. LVCSR Switchboard April 1996 Evaluation Report. In *The LVCSR Hub 5 Workshop*, April 29 - May 1 1996.

[15] Michael Finke. Personal Communication. unpublished, 1997.

[16] P. Finke, M. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal. The Karlsruhe-Verbmobil Speech Recognition Engine. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[17] Eric Fosler. Personal Communication. unpublished, 1996.

[18] J.L. Gauvain, L. Lamel, G. Adda, and M. Adda-Decker. The LIMSI Nov93 WSJ System. In *ARPA Speech and Natural Language Workshop*, March 1994.

[19] J.L. Gauvain, L. Lamel, and M. Adda-Decker. Developments in Large Vocabulary Dictation: The LIMSI Nov94 NAB System. In *ARPA Spoken Language Systems Technology Workshop*, January 1995.

[20] L. Gillick and Y. Ito. Confidence Estimation and Evaluation. LVCSR Hub5 Workshop Presentation, October 1996.

[21] A. Hauptmann and H. Wactlar. Indexing and Search of Multimodal Information. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1997.

[22] A.G. Hauptmann, M.J. Witbrock, A.I. Rudnicky, and S. Reed. Speech for Multimedia Information Retrieval. In *The User Interface Software Technology Conference*, November 1995 1997.

[23] X.D. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, K.F. Lee, and R. Rosenfeld. The SPHINX-II Speech Recognition System: An Overview. *Computer Speech and Language*, 2:137–148, 1993.

[24] X.D. Huang, F. Alleva, M.Y. Hwang, and R. Rosenfeld. An Overview of the SPHINX-II Speech Recognition System. In *ARPA Human Language Technology Workshop*, March 1993.

[25] Don R. Hush and Bill G. Horne. Progress in Supervised Neural Networks. *IEEE Signal Processing Magazine*, pages 8 – 39, January 1993.

[26] M. Hwang, R. Rosenfeld, E. Thayer, R. Mosur, L. Chase, R. Weide, X. Huang, and F. Alleva. Improving Speech Recognition Performance via Phone-Dependent VQ Codebooks and Adaptive Language Models in Sphinx-II. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1994.

[27] Mei-Yuh Hwang. *Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition.* PhD thesis, School of Computer Science, Carnegie-Mellon University, December 1993.

[28] M.Y. Hwang and X.D. Huang. Shared-Distribution Hidden Markov Models for Speech Recognition. Technical report cmu-cs-91-124, Carnegie Mellon University, April 1991.

[29] P. Jeanrenaud, M. Siu, and H. Gish. Large Vocabulary Word Scoring as a Basis for Transcription Generation. In *Proceedings of Eurospeech*, pages 2149–2152, 1995.

[30] F. Jelinek. Self-Organized Language Modeling for Speech Recognition. In A. Waibel and K-F. Lee, editors, *Readings in Speech Recognition.* Morgan Kaufman Publishers, San Mateo, CA, 1990.

[31] S.M. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35:400–401, March 1987.

[32] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, 1995. IEEE.

[33] F. Kubala. Fill Me In – Description of NAB DATA. In *DARPA Speech and Language Workshop*, January 1995.

[34] K.F. Lee and S. Mahajan. Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition. Technical Report CMU-CS-89-100, Carnegie Mellon University, January 1989.

[35] C. Leggetter and P. Woodland. Speaker Adaptation of HMMs Using Linear Regression. Technical Report TR 181, Cambridge University, Cambridge, UK, June 1994.

[36] A. Ljolje, M. Riley, D. Hindle, and F. Pereira. The ATT 60,000 Word Speech-to-Text System. In *ARPA Spoken Language Systems Technology Workshop*, pages 162–165, January 1995.

[37] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall, 2-6 Boundary Row, London SE1 8HN, UK, 1989.

[38] Ravishankar Mosur and Maxine Eskanzi. Personal communication, February 1997.

[39] H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub. Large-Vocabulary Dictation Using SRI's Decipher Speech Recognition System: Progressive Search Techniques. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 319–322, April 1993.

[40] H. Murveit, P. Monaco, V. Digalakis, and J. Butzberger. Techniques to Achieve an Accurate Real-Time Large-Vocabulary Speech Recognition System. In *ARPA Human Language Technology Workshop*, pages 368–373, March 1994.

[41] C. Neti, S. Roukos, and E. Eide. Confidence Measures Based Search Strategy for Continuous Speech Recognition. LVCSR Hub5 Workshop Presentation, October 1996.

[42] L. Nguyen, T. Anastasakos, F. Kubala, C. LaPre, J. Makhoul, R. Schwartz, N. Yuan, G. Zavaliagkos, and Y. Zhao. The 1994 BBN/BYBLOS Speech Recognition System. In *ARPA Spoken Language Systems Technology Workshop*, pages 77–81, January 1995.

[43] Y. Normandin, D. Bowness, R. Cardin, C. Drouin, R. Lacouture, and A. Lazarides. CRIM's November94 Continous Speech Recognition System. In *ARPA Speech and Natural Language Workshop*, pages 153–155, January 1995.

[44] J. Odell, V. Valtchev, P. Woodland, and S. Young. One Pass Decoder Design for Large Vocabulary Recognition. In *ARPA Human Language Technology Workshop*, 1994.

[45] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, A. Martin, and M. Przybocki. 1994 Benchmark Tests for the ARPA Spoken Language Program. In *ARPA Spoken Language Systems Technology Workshop*, pages 5–38, January 1995.

[46] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, and M. Przybocki. 1993 Benchmark Tests for the ARPA Spoken Language Program. In *ARPA Speech and Natural Language Workshop*, pages 15–40, March 1994.

[47] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, A. Martin, and M. Przybocki. 1995 Hub-3 NIST Multiple Microphone Corpus Benchmark Tests. In *ARPA Speech Recognition Workshop*, February 1996.

[48] D. Paul. An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model. Technical Report 930, Lincoln Laboratory, July 1991.

[49] D.B. Paul and J.M. Baker. The Design for the Wall Street Journal-based CSR Corpus. In *DARPA Speech and Language Workshop*, San Mateo, CA, Feb 1992. Morgan Kaufmann Publishers.

[50] Mosur K. Ravishankar. *Efficient Algorithms for Speech Recognition.* PhD thesis, Computer Science Department, Carnegie Mellon University, May 1996.

[51] F. Richardson, M. Siu, and H. Gish. Confidence Scoring for the 1996 Spanish CallHome Evaluation. LVCSR Hub5 Workshop Presentation, October 1996.

[52] Ze'ev Rivlin, Michael Cohen, Victor Abrash, and Thomas Chung. A Phone-dependent Confidence Measure for Utterance Rejection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 515–517, May 1996.

[53] Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1994. Ph.D. thesis also published as Technical Report CMU-CS-94-138.

[54] Ronald Rosenfeld. The CMU Statistical Language Modeling Toolkit, and its use in the 1994 ARPA CSR Evaluation. In *Proc. ARPA Spoken Language Technology Workshop*, Jan 1995.

[55] Kristie Seymore and Roni Rosenfeld. Personal Communication. unpublished, 1997.

[56] S. S. Stevens and J. Volkmann. The Relation of Pitch to Frequency:a Revised Scale. *The American Journal of Psychology*, 53:329–353, 1940.

[57] B. Suhm, B. Myers, and A. Waibel. Interactive Recovery from Speech Recognition Errors in Speech User Interfaces. In *The International Conference on Spoken Language Processing*, volume 2, pages 861–864, October 1996.

[58] Bernhard Suhm. Personal communication, January 1997.

[59] W.N. Venables and B.D. Ripley. *Modern Applied Statistiscs with S-Plus.* Springer-Verlag, New York, USA, 1994.

[60] Wayne Ward. Personal Communication. unpublished, 1997.

[61] Withgott and Chen. *Computational Models of American Speech.* 1995.

[62] P. Woodland, M. Gales, D. Pye, and V. Valtchev. The HTK Large Vocabulary Recognition System for the 1995 ARPA H3 Task. In *ARPA Speech Recognition Workshop*, February 1996.

[63] P. Woodland, C. Leggetter, J. Odell, V. Valtchev, and S. Young. The Development of the 1994 HTK Large Vocabulary Speech Recognition System. In *ARPA Spoken Language Systems Technology Workshop*, pages 104–109, January 1995.

[64] P. Woodland, J. Odell, V. Valtchev, and S. Young. The HTK Large Vocabulary Continuous Speech Recognition System: An Overview. In *ARPA Speech and Natural Language Workshop*, pages 98–101, March 1994.

[65] Sheryl R. Young. Recognition Confidence Measures: Detection of Misrecognitions and Out-of-Vocabulary Words. Technical Report No. CMU-CS-94-157, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 15232, USA, 1994.

[66] T. Zeppenfeld, M. Finke, K. Ries, M. Westphal, and A. Waibel. Recognition of Conversational Telephone Speech using the Janus Speech Engine. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997.