Unsupervised segmentation of audio speech using the Voting Experts algorithm

by

Matthew Adam Miller

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Alexander Stoytchev, Major Professor
Giora Slutzki
Pavan Aduri

Iowa State University

Ames, Iowa

2009

DEDICATION

To Cat.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to acknowledge Paul Cohen and Wesley Kerr from the University of Arizona for generously sending us the source code for the original VE algorithm, and Paul Cohen for his helpful comments about our algorithm. I would also like to thank Richard Aslin from the University of Rochester for providing us with the stimulus streams used in the original Saffran *et al.* experiments, and Peter Wong for developing a system to visualize induced breaks in an audio sequence. Finally, I must acknowledge my advisor Alex Stoytchev, who was the co-author on all of this research, and helped formulate many of the ideas and experiments.

ABSTRACT

In this thesis I suggest and evaluate an algorithm for the unsupervised segmentation of audio speech streams. Specific attention will be paid to the developmental psychology of human infants, who learn to perform this task at an early age. The goal will be to both suggest an algorithm inspired by the human distributional segmentation mechanism, and to evaluate the performance of that model on acoustic speech. I will focus on the audio domain, in contrast to a great body of previous work devoted to the unsupervised segmentation of text. The algorithm presented is used to reproduce a famous series of infant experiments, and shown to perform similarly to the children. It is also used to segment a large audio corpus, which it does with accuracy significantly better than chance. Finally, improvements to the acoustic model and segmentation algorithm are outlined, implemented and tested, demonstrating the potential for future development of the system.

CHAPTER 1. INTRODUCTION

Spoken human language contains no analogue to the spaces placed between written words. The pauses that do exist in audio speech appear between phrases, when the speaker takes a breath, or when the airflow is stopped in the pronunciation of certain consonants. The sounds that are separated by these pauses are rarely composed of a single word, and there are no universal markers to indicate where those single words might be [Klatt (1979)]. However, when we hear our native language, we hear discrete words. We unconsciously break the stream into its constituents, rendering it comprehensible. This is possible because we know the language, and are familiar with the large lexicon of words we might expect to hear. When confronted with a novel word, we need only segment the words before and after it to identify it as a brand new token.

Infants, however, do not share this luxury. They must learn to segment their mother's tongue from scratch. Every word is a novel word, and their lexicon starts off empty. Fortunately, human beings have an apparently innate ability to segment continuous spoken speech into words, and that ability is present in infants as young as 8 months old. Apparently, they can perform this task without any feedback or other salient cues as to the locations of word breaks [Saffran et al. (1996, 1999)].

There is, however, moderate disagreement regarding precisely how children learn to perform this task. And while there is a consensus that they must use some combination of various techniques, the relative importance of those techniques is not well known.

Specifically, there is uncertainty regarding infants' use and reliance upon the distributional cues of speech for identifying words and word boundaries [Jusczyk (1999); Mattys et al. (2005)]. When we say "distributional cues" we mean the statistical properties of the sound sequences

that make up spoken language. In any particular language there are certain sound sequences that occur more often than others. Psychologists in the "statistical learning" community have suggested that infants learn to recognize these statistical regularities, and use them to break speech into words [Harris (1955); Saffran et al. (1996)].

However, distributional cues are not the only information available when learning to perform this task. Parents often use single, isolated words when speaking to infants [Brent and Siskind (2001)]. This information is clearly advantageous for anyone trying to learn to segment a novel language. Moreover, there are also prosodic or "metrical" cues, allophonic variations, and phonotactics. Infants have been shown to be sensitive to each of these cues during different periods of their development, and they almost certainly use a combination of all of these factors to learn to identify word breaks [Jusczyk (1999)]. However, it isn't clear how these cues interact, and the precise order in which children learn to recognize them. I will discuss each of these in detail in Chapter 2.

I suggest, as others have [Gambell and Yang (2008)], that the very first cues the child must use are the distributional ones. These are not language specific, and require no prior knowledge of the language to learn. However, it is still unclear which statistics the children pay attention to. Which ones are the most useful for segmentation? Can we model this process, and perhaps use this model to replicate the behavior of infants? The central contribution of this thesis will be to suggest such a model, use it to perform segmentation experiments, evaluate its performance, and demonstrate how it can be extended and improved.

The goal of this research is not to create a complete model of the entire human speech segmentation process. Such a mechanism certainly requires components of statistical learning, lexical construction, metrical, allophonic and phonotactic learning, as well as exploratory interaction with competent language users. Human infants certainly make use of all of these techniques. Instead, this work focuses on the narrow problem of beginning the process. Infants start without any language specific knowledge, and yet somehow learn to break language streams into words. However they do this, they must start from the data - using only the language that they hear to begin to find word breaks. How they leverage this initial knowledge

into better and better segmentation is beyond the scope of this work. The focus here is how they take that initial step. Accordingly, the goal is not to induce perfect segmentation in the given audio streams. Instead, it is to induce segmentations whose quality is substantially greater than chance, and that might be used to bootstrap further word learning.

When discussing algorithms that mimic the abilities of human beings, it is important to be clear about the interplay between accurate cognitive modeling and useful practical results. Those are two entirely separate branches of research, and they rarely coincide. This work draws inspiration from the developmental psychology of infants, but the goal is not a biologically or psychologically accurate facsimile. The goal is to follow the lead provided by an enormous body of psychological research, and to work toward practical segmentation algorithms that can perform similar tasks.

Infants have shown us that acoustic segmentation is possible, and their behavior provides a road map for how it might be done. This map is made explicit by the order in which they grow sensitive to different acoustic cues, and the types of audio streams they are able to segment. This structure defines a strategy that is proven to work, since children do, in fact, learn to parse language. The algorithms presented in this thesis will be tested on some of the same datasets used to test children. The purpose of those tests is not to prove that the cognitive model is accurate, but instead to verify that the inspiration drawn from children carries through to practical results. That is, the algorithms are designed to reproduce the segmentation behavior of infants by using methods and acoustic cues similar to those that infants use. This is interesting because the segmentation ability of infants is amazing, not because it provides an accurate cognitive model. We are trying to reproduce the same results, or at least take one small step in that direction. Accordingly, the proposed algorithms will be expected to perform as the children do, to demonstrate that we are on the right track.

Furthermore, another goal of this research is to go beyond historic attempts at unsupervised natural language segmentation, and actually try to segment words from audio streams. Previous models of infant speech segmentation have dealt almost exclusively with text transcripts. This thesis is, to my knowledge, the first real attempt to apply one of these models to

actual acoustic speech. I suggest that difficulties and challenges inherent to real audio segmentation are not trivial details, but a necessary and invaluable part of the automatic processing of real world data. Algorithms that cannot handle the analog nature of sensory streams, or the associated noisiness of the observations, are not well suited to model this process.

Also, the practical usefulness of unsupervised automatic text segmentation is somewhat questionable. There do exist certain written languages, like Chinese, that do not contain breaks between words. In that case, text segmentation is an integral part of other natural language processing techniques. However, these algorithms can be completely supervised, provided with large dictionaries of legal words, and otherwise engineered to accomplish the desired task. The only reason an unsupervised algorithm might be needed is to simulate infant learning, and to uncover similarly powerful algorithms that can be used to mine data from noisy, analog sensory signals. Therefore, while unsupervised text-based algorithms might begin to model the cognitive process of infants, they miss the most interesting point - that children are remarkably good at extracting patterns from a messy world. This thesis focuses on segmenting real audio for precisely this reason, to face the same challenges as the infants do, and to produce segmentation algorithms that have a chance at being useful in the future.

1.1 Research Questions

The organization of this thesis is centered around the following research questions.

1. What are the first cues that infants use to begin segmenting language?
2. Are there any existing segmentation algorithms that utilize similar cues to break sequences? If so, which is the best for our purposes?
3. Given such an algorithm, how can it be used to segment acoustic speech?
4. Does it perform similarly to infants under similar experimental conditions?
5. Can the algorithm be used to induce breaks in naturally occurring speech?

6. Can the segmentation quality be improved through improvements in the acoustic modeling of speech streams?

7. Can the segmentation algorithm itself be extended and improved?

The first two of these questions are answered in Chapter 2, through a careful review of psychological and unsupervised segmentation literature. Questions 3, 4 and 5 are answered in Chapter 3, which outlines a method for the application of the Voting Experts segmentation algorithm to acoustic speech, and uses it to replicate a famous series of infant experiments, as well as to segment an audio book. Question 6 is answered in Chapter 4, which introduces a more sophisticated acoustic model and uses it to improve the algorithm's performance on some of the experiments from Chapter 3. Finally, question 7 is addressed in Chapter 5, where Voting Experts itself is extended, improved and thoroughly tested. Chapter 6 summarizes the body of the work, draws some conclusions and outlines the direction of possible future research on this topic. Notably, questions regarding the more general problem of the automatic segmentation of sensory or time series input are conspicuously absent from this document. This is certainly an open problem, and the contributions of my work may have application outside the realm of speech processing. However, these questions are outside the scope of this thesis, and will not be addressed in detail.

CHAPTER 2. REVIEW OF LITERATURE

There are two main areas of previous research that directly relate to this thesis. The first is the developmental psychology of speech segmentation. There exists vast amounts of literature on this topic, and an extremely detailed analysis of the field is beyond the scope of this work. However, I will endeavor to summarize the main conclusions, and point out some particularly relevant experiments.

The second area of related work consists of previous unsupervised segmentation algorithms. The substantial majority of these algorithms are designed to run either on text, or textual representations of phonemic transcripts of speech. That is, these algorithms typically do not work on audio streams. For this reason, both in the review of the psychological literature and in the discussion of previous algorithms, it will be assumed that all "segmentation" experiments were performed using text based phonetic transcripts of speech. If this is not the case, it will be explicitly mentioned.

One might argue that this is an over-simplification of the problem. It is far more difficult to learn to segment noisy audio streams than error free phonetic transcripts. If it were possible to build unsupervised acoustic models that accurately transcribed human speech into phonemes or syllables, then this might be a non-issue. However, this is currently not possible. The acoustic models in chapter 3 will have to deal explicitly with this problem. However, psychologists are often not concerned about the particulars of audio processing. It is also significantly easier to write algorithms that process textual phonetic transcripts instead of raw audio speech. Both of these factors combine to produce a great deal of work on segmenting text, and not a lot on segmenting real audio.

2.1 Psychology of Speech Segmentation

It's generally believed that children begin learning to segment speech when they are around 7 months old, and that they gain proficiency over the course of many months. By the time they are moderately skilled language users, they are most likely using a complex combined strategy to induce word boundaries. That is, there are several different cues present in language streams that can be used to find word boundaries, and children seem to be sensitive to many of them [Jusczyk (1999); Mattys et al. (2005)]. Unfortunately, most of them require at least moderate segmentation proficiency before they can be employed. That is, they are methods of improving segmentation as opposed to learning to do it from scratch. This current work is focused on learning the task from scratch, so special emphasis will be placed on those strategies that can be used to do so. This is not to say that the other methods are not important. In fact, it's likely impossible to adequately segment speech without using these bootstrapping methodologies. Those techniques which allow the process to get off the ground may not carry the process very far. However, it is important to start at the beginning, and therein lies the justification for the focus of this research.

The following sections will discuss the different methodologies that children might use to segment language. Each one of them is based on some property of human language - either regarding its general structure, or the way we happen to speak it. These properties fall into two broad categories: language dependent and language independent.

Language dependent properties are those that vary from particular language to particular language. These sort of rules will be less important for the purposes of this thesis, since, in most cases, they cannot be learned until the language is segmented. It is therefore difficult, or even impossible to use them to begin the segmentation process. Language independent properties are shared by all known human languages. For instance, all human languages can be physically pronounced by human beings. This seems trivial, but it heavily constrains the sort of sounds that can appear in language. These properties are much more useful, since they typically do not require knowledge of a specific language in order to learn. Therefore they are much more likely to provide an appropriate starting point for segmentation.

2.1.1 Statistical Learning

The idea that infants use statistical cues to segment speech streams has a long history [Chomsky (1955); Harris (1955); Hayes and Clark (1970); Wolff (1977); Pinker (1984); Goodsitt et al. (1993)]. Specifically, the theory is that they use the transitional probabilities between syllables as an indicator of word boundaries. Let the transitional probability from syllable A to syllable B be defined as

$$\Pr(A \rightarrow B) = \frac{\Pr(AB)}{\Pr(A)}$$

It stands to reason that syllables that appear together inside of a word would have a higher Transitional Probability (TP) than those that do not. Therefore, the argument goes, the transitional probabilities between syllables inside of words should be high, but the TP between syllables that cross a word boundary should be low. Consider the utterance "pre-tty-ba-by." According to this theory, the transition probability $\Pr(pre \rightarrow tty)$ should be higher than $\Pr(tty \rightarrow ba)$. This seems intuitive, since the syllable sequences inside of words appear together whenever the word is spoken, and the syllable transition between words does not. This should be a universal property of any sequence that is composed of unique lexical units (words), which are themselves composed of tokens (syllables). The theory, then, is that infants learn to recognize both common and uncommon syllable transitions, and postulate word breaks at local minima in the transition probabilities of the speech stream.

2.1.1.1 Infant Segmentation Experiments

Human infants have been shown to be remarkably sensitive to these statistical cues. In a now-famous series of experiments Saffran *et al.* showed that 8-month-old infants are capable of segmenting an audio stimulus stream based solely on these transition probabilities [Saffran et al. (1996)]. Several additional studies have demonstrated that adults have the same ability, not only in the auditory but also in the visual domain [Saffran et al. (1997, 1999); Fiser and Aslin (2002); Kirkham et al. (2002)].

Saffran and colleagues performed their experiments using an artificial language made up of the 4 nonsense words *tupiro*, *golabu*, *bidaku*, and *padoti*. These words are constructed from 12 unique syllables, so that the concatenation of no two words can produce another word in the language. They artificially generated their stimulus streams using a voice synthesizer such that the streams contained no pauses, variations in emphasis or meter, or any other cues as to the word boundaries. The streams simply consisted of the nonsense words concatenated in random order, and spoken in an even, monotonous tone.

The only clue as to the proper break locations of the stream were the transition probabilities between syllables. The transition probability between syllables inside of words was always 100%. For instance, the syllable "tu" was followed by "pi" with probability 1, since the only place those two syllables existed in the language was in the word *tupiro*. However, the syllable "ro" (at the end of the word) could be followed by one of four possible syllables (the beginning of the next word): "tu," "go," "bi," or "pa." Therefore, the transition probability between words dropped to 25%. Saffran *et al.* postulated that if children were using statistical learning to segment speech, they would be able to decipher these stimulus streams.

To test this hypothesis, they played this stimulus stream to 8-month-old infants for 2 minutes. After the children were acclimated to the sound, they played them a single repeated word. Some of the children heard a word from the original language. Others heard a novel word that was generated from the same syllables, but was not present in the original stimulus stream. The infants spent a significantly longer period of time paying attention to the novel word than the familiar one. This demonstrates that the children were able to segment the original stimulus stream and learn the nonsense words after only 2 minutes of exposure. They were also able to recognize a novel word, and were confused and interested enough to pay attention to it.

The results of these experiments were taken as evidence that human infants really do pay attention to the transitional probabilities between syllables, and that they use them to segment audio speech. However, that's not really what these experiments showed. They showed that infants can segment audio speech using *some kind* of statistical model, and that it is powerful

enough to work on the stimulus stream they were presented. Dips in inter-syllable transition probability were the simplest cue that they could have used to segment the sequence, but virtually any sophisticated model should have picked up this very simple pattern. And there is significant evidence to suggest that infants, in fact, are not using TPs to do this.

Most dramatically, multiple studies have shown that the direct application of the TP strategy performs poorly when used to segment phonetic transcripts of speech [Cairns and Shillcock (1997); Gambell and Yang (2008)]. This exposes several of the weaknesses of the traditional statistical learning approach. First of all, a very high percentage of common words contain only one syllable. It is therefore impossible for there to be a TP valley on both sides of the word. Moreover, the original conclusion that word-internal transitions should have higher probabilities than word-external ones is not always true in practice. Often, the last syllable of one word and the first syllable of the next happen to form a perfectly common pair. Similarly, many words contain syllable combinations that are, in general, rare (perhaps only appearing in a handful of words). The difference in single-syllable TP inside of and between words is more of a trend than a reliable rule.

Because of these shortcomings, a different algorithm was chosen for the experiments presented here. However, the insight that motivates this theory will still play a crucial role. It will simply take the form of a much more powerful model. And we should not forget the contribution of the infant experiments. They demonstrate that we have some kind of mechanism for statistical segmentation. These cues can be calculated right from the data, and learned without supervision or prior domain knowledge. This positions this segmentation mechanism, whatever it might be, as a likely starting point of the language learning process.

2.1.2 Isolated Words

Isolated words are obviously useful when learning a new language, to infants and adults alike. And it is well known that infants learn the words that their parents use in isolation much faster than other words [Brent and Siskind (2001); Bortfeld et al. (2005)]. It has also been suggested that children use isolated words to bootstrap the learning of novel word boundaries

through a process called "subtraction" [Peters (1983); Pinker (1984)]. Suppose a child already knows the words "ball" and "red." Then if she hears the sentence "The ball is red," she can subtract out the words she knows, leaving "The ... is ...". This way she can learn two novel words without needing to ever hear them in isolation.

However, children must still solve the problem of differentiating between isolated word utterances and short phrases, since utterance length is not a consistent predictor [Gambell and Yang (2008)]. For instance, the phrase "I am" is shorter than the single word "lasagna." In order to use isolated words to begin to learn speech, children must be able to identify when isolated words are spoken.

Approximately 9% of words in infant directed speech are spoken in isolation [Brent and Siskind (2001)]. This is disproportionately high compared to normal speech, but even so, isolated words make up a small minority of the language that infants hear. Short phrases consisting of two or three words are also very common. Without any mechanism by which to tell the difference between isolated words and short phrases, the bootstrapping can never get off the ground. This would seem to necessitate some other learning mechanism that precedes and augments isolated word recognition - to bootstrap the bootstrapping.

So while isolated word recognition is clearly useful and necessary in the early stages of language learning, it is not sufficient to account for the entire process. In fact, I suggest that it is precisely this insufficiency that creates the need for distributional segmentation.

2.1.3 Allophonic Variation

The pronunciation of a phoneme varies based on its position within a word [Krakow (1999)] and this may help children find word boundaries [Church (1987)]. For instance, the sound made by a 't' is usually aspirated at the beginning of a word, but not at the end. Notice the difference in pronunciation between the 't' in "tall" and the 't' in "bat." There are many such regularities, particularly regarding the amount of co-articulation between phonemes. That is, how much the pronunciation of one phoneme affects the pronunciation of the next. This affect is noticeably stronger between phonemes inside of a word, and weaker when the phoneme pair

crosses a word boundary [Liberman et al. (1967)]. If a child was sensitive to such cues, it would certainly help her find word boundaries.

It has been pointed out, however, that this knowledge cannot be innate to the child [Gambell and Yang (2008)], since allophonic variation is language dependent. These pronunciation rules must be learned through exposure to the language, and therefore require significant segmentation proficiency before they get off the ground. As with many of these cues, they certainly play a role in the language learning process, but they could not be the initial starting point. In fact, it seems as though younger infants pay less attention to these cues, while older ones pay more [Jusczyk et al. (1999)]. This suggests that children become sensitive to articulatory variations later in their development than both distributional cues and isolated words.

2.1.4 Metrical Cues

Metrical or "prosodic" variation refers to the emphasis we place on certain syllables within each word. In English, most words are "stress initial," meaning that the emphasis is on the first syllable. However, this is not universally true. For instance, the word "complete" is stress final. And some homonyms are even distinguished by their stress. A "reject" (stress initial) is a person that we all "reject" (stress final). Several studies have demonstrated that infants develop a strong affinity to the metrical pattern of their native language [Jusczyk et al. (1993a)].

This pattern, however, varies between languages. While English is predominantly stress initial, others are predominantly stress final. A child would need to learn a significant number of words in a language before the typical metrical pattern could be identified. Once again, this would suggest that this is not a ground-floor strategy, but rather a bootstrapping add-on that improves later performance.

Gambell and Yang have suggested an interesting alternative to the metrical segmentation strategy [Gambell and Yang (2008)]. Perhaps children always assume that one syllable per word is emphasized. This is generally true of human language (with a few exceptions), and

would allow the child to begin segmentation without any prior knowledge. This, of course, would not help specify the exact location of word breaks in many cases, but it certainly would narrow down the possibilities. Also, a large number of common words consist of only one syllable. This strategy would give the children those words for free.

It is not known whether children really do use this strategy, but it would seem advantageous to do so. Metrical cues are certainly informative, and infants do make good use of them, but I will not be addressing them further in this work. It is still speculative whether this propensity is innate or learned in children. Metrical patterns certainly vary from language to language, so those must definitely be learned. Whether infants assume that there is one stressed syllable per word remains to be seen. Even if they do, this would not provide specific break locations, but only clues to help narrow down the possibilities. Perhaps such cues could be used to augment future acoustic segmentation models, but they do not seem promising as an initial starting point.

2.1.5 Phonotactic Constraints

A syllable can be decomposed into onset and rime, and rime can be further decomposed into a nucleus vowel and a coda. Both the onset and the coda consist of consonant clusters, which are one or more consonant sounds. However, not all possible consonant clusters are "legal" in onsets and codas in English. For instance, a syllable cannot start with two guttural stops. Therefore the consonant cluster "pg" would not be a valid onset. In fact, there are not a lot of consonant clusters that are allowed to serve as onsets [Halle (1978)].

The *phonotactics* of a language are the rules for the legal combination of phonemes into syllables. These rules are not universal, and vary greatly between languages. This means, once again, that children must learn them from experience. However, several studies have shown that children as young as 9 months old have already learned the phonotactic constraints of their native language [Jusczyk et al. (1993b, 1994); Mattys and Jusczyk (2001)].

Phonotactic knowledge can be used to induce breaks in an audio stream whenever a consonant cluster violates one of the constraints. That cluster must actually mark a syllable

boundary. For instance, in the sentence "help the chap get out of the car," the consonant cluster "pg" appears between "chap" and "get." Even if a child didn't know the word "chap," she could use phonotactic knowledge of English to induce a break.

Unfortunately, phonotactic rules can only be learned from segmented words. I will discuss a handful of algorithms that attempt to use utterance boundaries to learn the phonotactics of a language. However, they have proven rather poor at finding word boundaries from scratch, but most useful in the improvement of other segmentation algorithms.

2.1.6 Conclusions

One particular cue stands out as the most likely starting point for infant speech segmentation. Statistical learning requires no previous domain knowledge, no lexicon and no other data besides the raw acoustic stream. This is not true of any of the other strategies. Metrical, allophonic and phonotactic cues are all language specific, and require proficient segmentation abilities before they could be learned. The only other candidate is isolated words. It's true that infants respond to isolated words at a very early age. However, some method is needed to tell the difference between short phrases and whole words. Furthermore, infants would require large amounts of data before enough isolated words had been learned to allow for useful segmentation of novel sentences. Therefore, we can't help but conclude that some sort of distributional mechanism is most likely responsible for the initial segmentation. Perhaps it is accompanied by isolated word recognition, or perhaps that ability is learned soon after. Either way, statistical learning seems to be the most promising avenue for algorithm development. Therefore, in this thesis, a segmentation algorithm is chosen that uses statistical information to induce breaks in sequences.

2.2 Segmentation Algorithms

There have been a wide range of algorithms suggested for the unsupervised segmentation of natural speech. These strategies vary both in their goals and their methodology. Some draw heavy inspiration from developmental psychology, and attempt to build realistic models of the

process. Others merely focus on reproducing the results, without considering whether their algorithms are "biologically plausible." The algorithms also differ regarding which segmentation cues they regard as important, and how they define successful segmentation. Is it good enough to merely indicate the indices of breaks, or is it also important to build a lexicon of recognized words?

I will attempt to outline a representative set of previous segmentation algorithms, and categorize them into several main approaches. Finally, I will describe the Voting Experts algorithm, which will be used in all of the segmentation experiments in this thesis.

2.2.1 Probabilistic Models

Several segmentation algorithms attempt to learn a generative model of the language stream. Typically, these models specify both a lexicon of possible words and probabilistic rules for the use of those words. During training, some sort of lexicon is extracted and used to parse the stream based on some rule or criteria. The selection of model, lexical structure and parsing rules differentiates the individual algorithms.

2.2.1.1 MDL

One of the first of these models was due to Carl De Marcken [de Marcken (1995)], and was based on finding the minimum description length of the given language strings. De Marcken used the MDL principle to learn a probabilistic grammar with which to parse a given corpus. Production rules were added to the grammar whenever their addition reduced the total number of bits needed to represent the corpus. The result of the algorithm was a hierarchy of language segmentation - the highest level corresponding to phrases or sentences, and the lowest level corresponding to each character in the text. The argument was that this multi-level structure represented each level of linguistic segmentation, from letter to morph to word to phrase. Unfortunately, there is no way to determine which level in the hierarchy corresponds to "words." In fact, that level varies from word to word. For that reason, this algorithm is not very applicable to this research project, however useful it might be for text compression.

2.2.1.2 Bayesian Models

The most well known Bayesian model of language segmentation was due to Michael Brent [Brent (1999)], and was called $INCDROP$. $INCDROP$'s abstract generative model produced an utterance using the following 5 steps:

1. Choose an integer n, which represents the number of words in the lexicon
2. Choose n distinct strings which represent the n words W_i. Call this lexicon $L = \{W_1, ..., W_n\}$. Let $W_0 = \$$, the utterance boundary marker.
3. Choose a function $f : \{0, ..., n\} \rightarrow \{0, 1, ...\}$ where $f(i)$ represents the number of times word W_i appears in the sequence being generated.
4. Let m be the total number of words in the sequence being generated. Select an ordering function $s : \{1, ..., m\} \rightarrow \{1, ..., n\}$ that maps each location in the sequence to a corresponding word in L.
5. Concatenate the word sequence $w_1...w_m$ without removing utterance boundaries to produce an unsegmented corpus.

Notice that in this model, utterance boundaries are included. In order to perform segmentation several simplifying assumptions were made. Specifically, a uniform prior was assumed over all functions f and all word orders. That is, each subsequent word in the sequence was assumed to be independent of its neighbors. This model can be used to calculate the likelihood of a parse of a given unsegmented corpus (with utterance boundaries included). More importantly, the model can be used to find the most likely parse of a given utterance.

Brent's algorithm MBDP-1 does just that. It works by incrementally segmenting one utterance at a time. Given an utterance, it uses dynamic programming to find the maximum likelihood parse using its current lexicon L and word counts. Then, the word counts are updated based on the parse, and any unparsed character sequences are added to the lexicon as a new word. Initially, when the lexicon is empty, the entire utterance is added as a "word." However, utterances containing short phrases and isolated words allow the algorithm to get

off the ground, and are used to break apart larger phrases. For instance, if the algorithm is presented with the utterance "look" and then "lookhere," it would first add "look" to the lexicon, and then use it to parse "lookhere" into "look" and "here". Sometimes short phrases can also be used in the same way as isolated words. For instance if the algorithm was presented the utterances "theredball" and then "grabtheredball," it would use the first phrase to parse out the word "grab" from the second. As more and more utterances are parsed the lexicon grows, and eventually "correct" lexical elements begin to dominate over the initial incorrect ones. The precision and recall of MBDP-1 settle above 80% when segmenting phonetic transcripts of child directed speech [Brent (1999)].

There have been several other models whose structure and performance were similar to Brent's. Specifically, Anand Venkataraman proposed an online segmentation algorithm whose results were extremely similar [Venkataraman (2001)]. Venkataraman's algorithm made use of bi-gram and tri-gram word models in order to approximate contextual information in the model, however the results were shown to be insensitive to that addition. More recently, Sharon Goldwater has suggested a more sophisticated model that treats word production as a Dirichlet process [Goldwater et al. (2006)]. Her results were slightly superior to Brent and Venkataraman.

These approaches are all somewhat promising, and have demonstrated the best performance at segmenting phonetic speech transcripts of speech. However, all of these models leverage utterance boundaries and isolated words to begin learning a lexicon. While this is certainly part of the infant strategy, Saffran's experiments have demonstrated that children do not need these cues to begin segmenting speech streams. Moreover, it is difficult to see how these algorithms might be applied to acoustic speech segmentation. These models require reliable string matching to induce parses on utterances. Perhaps some sort of sophisticated pattern matching may allow similar methods to be used on audio, but it is not straightforwardly obvious how this could be done.

2.2.2 Phonotactic Models

Phonotactic segmentation is based on the idea that there are only a small number of legal phoneme sequences that can compose the onset or coda of a syllable in a given language. These rules can be expressed statistically, representing the probability that a given sequence of phonemes precedes or crosses a word boundary. It is easy to learn these probabilities from a segmented corpus, and supervised phonotactic segmentation algorithms perform very well [Cairns and Shillcock (1997)]. It is more difficult, however, to train these models on unsegmented text [Cairns and Shillcock (1997); Christiansen et al. (1998); Brent and Cartwright (1996)]. All of these methods use utterance boundaries in order to learn the phonotactic models. That is, the probability that a bi-gram or tri-gram of phonemes precedes a *word boundary* is approximated by the probability that that bi-gram or tri-gram precedes an *utterance boundary*. These models perform very poorly, demonstrating that phonotactic knowledge is best used to augment or improve a more primary segmentation strategy.

More recently, several studies have used phonotactic knowledge along with other segmentation strategies to improve performance. The WordEnds algorithm combined phonotactic and lexical learning to obtain results competitive with other probabilistic model based approaches [Fleck (2008)]. More impressively, Blanchard and Heinz demonstrated how to include bi-gram and tri-gram phonotactic models of word boundaries into Brent's $INCDROP$ model [Blanchard and Heinz (2008)]. Essentially, they altered MBDP-1 such that the maximum likelihood parse of an utterance also accounted for the learned phonotactic rules, which were extracted from the lexicon. This addition improved the performance of the algorithm, demonstrating once again the usefulness of phonotactic knowledge. However, each of these experiments reinforces the point that phonotactic segmentation is not viable as an initial strategy for segmentation, but instead is tremendously useful for bootstrapping and improving a pre-existing method.

2.2.3 CELL

The CELL model is the only algorithm mentioned in this section that was designed to work with real audio streams [Roy and Pentland (2002)]. However, its purpose was not the

segmentation of natural language, but the association of grounded meaning with individual words. A by-product of this association was the ability to segment the words of interest out of the acoustic stream, but this was not the central focus. Accordingly, the model did not produce a neatly segmented sequence, but rather recognized a few words of interest.

CELL began by building a recurrent neural network to identify spoken phonemes using the TIMIT speech corpus [Garofolo et al. (1990)]. It then learned to associate specific phoneme sequences with objects from its environment, extracting "words" from the audio using MDL principles, and associating them with objects using mutual information statistics. Of the words extracted by the CELL architecture, 54% of them were segmented correctly. However, the model was not built to extract every word, but only those that corresponded to objects in its visual environment. Therefore it is not a suitable model for the purposes of this research. It represents an entirely different approach to initial language learning - associating meaning with a small set of extracted words instead of learning to segment a multitude of words whose meaning is initially mysterious. This is undoubtedly important in the language learning process, and its significance should not be underestimated. However, it is not strictly applicable to the segmentation task at hand, although the two should ultimately be part of the same mechanism.

2.2.4 Other Segmentation Algorithms

There are several other models for word segmentation that don't fit neatly into the other categories. For instance, there have been many attempts to train recurrent neural networks to segment phoneme sequences [Aslin (1996); Christiansen et al. (1998); Elman (1990); Cairns and Shillcock (1997)]. The networks were either used to predict utterance boundaries, or used to induce breaks when the next phoneme could not be predicted by the previous few. The performance of these methods was rather uninspiring, and neural network models soon fell out of fashion. There have also been several attempts to use local statistics to segment speech streams. For instance, Swingley (2005) used statistical clustering techniques to extract words from both English and Dutch. And Ando and Lee (2000) used n-gram frequency to segment

Japanese kanji, by postulating that high frequency n-grams were usually word-internal, while low frequency n-grams were word-external.

This is by no means a complete list of all segmentation algorithms or strategies. However, the vast majority of them fall into at least one of the categories that have been mentioned. I have also tried to provide a representative sampling of algorithms in those categories. However, none of the algorithms presented thus far admit a straightforward application to real audio, save the CELL model. The probabilistic models are attractive, given their theoretical rigor and the quality of their results. However, they rely too heavily on the ability to match elements in a lexicon to tokens in a stream - a non-trivial task when dealing with real audio. In other words, they rely on pristine data, and would be brittle when faced with noisy input. Conversely, phonotactic models might rely on modern speech recognition to identify short phoneme sequences, but are not well suited to begin the speech segmentation process. Even the CELL architecture does not directly address the segmentation problem, but instead focuses on learning the meaning of a handful of words. The rest of the models results are too poor to make them attractive candidates.

What is needed is a model of segmentation that makes local splitting decisions, that is efficient and can run on large amounts of data, that is theoretically sound and powerful, and that can be extended to work on acoustic streams. The last algorithm to be presented in this section is called Voting Experts, and I believe it to be the best algorithm for this task.

2.2.5 Voting Experts

Voting Experts (*VE*) is an algorithm for the unsupervised segmentation of discrete token sequences. It was first suggested by Paul Cohen [Cohen et al. (2007)], and has been shown to be proficient at segmenting large text corpuses from which all spaces and punctuation have been removed.

VE is based on the idea that natural "chunks" (in this case "words") exhibit two information theoretic properties. The first is **low internal information**. The second is **high boundary entropy**. For our purposes, when we say "information" we mean Shannon infor-

mation [Shannon (1951)], where Information and Entropy are defined as follows:

$$I(x) = -\log \Pr(x)$$
$$H(X) = \sum_{x \in X} \Pr(x) I(x)$$

A "chunk," in this case a short sequence s of letters, has low internal information precisely when the negative log of the probability of s is low. This value is minimized as the probability of s occurring approaches 1. So a chunk with low internal information is simply a sequence that occurs with high probability.

Conversely, the boundary entropy of a chunk s is the entropy of the subsequent token $t \in T$ following s. This is equivalent to the expected information content of the next token after the sequence s, which is defined as follows:

$$H_b(s) = \sum_{t \in T} \Pr(t|s) I(t|s)$$

The boundary entropy of a chunk s is maximized when all tokens t in the set of possible tokens T are equally likely to follow s. It is minimized when a single token $t \in T$ follows s with a probability of 1, and all other tokens follow s with probability 0. Boundary entropy is essentially a measure of how predictable the next token is.

If we consider a text corpus, we would expect the character sequences that make up words to occur with higher probability than character sequences that do not. We would also expect the characters to be highly predictable inside of words, but unpredictable at word boundaries. This illustrates the intuitive idea behind the *VE* model - that natural chunks should occur with high probability, and they should occur in many different contexts so that their boundaries are unpredictable. The word "yellow" occurs much more often in printed english than "lowhat," and can be found surrounded by many different words in phrases like "the yellow hat" or "my yellow car."

2.2.6 Voting Experts Implementation

The *VE* model uses two measures of information to induce a segmentation on a sequence. They are defined as follows. Let $\Gamma = \{e_1, e_2, ..., e_m\}$ be an alphabet and $s = (s_1, \ldots, s_n)$ be a

sequence where each element $s_i \in \Gamma$. The internal information of a subsequence $c = (s_j, \ldots, s_k)$ is given by $H_I(c) = -log(Pr(c))$. In other words, the internal information of a sequence is just the Shannon information of that sequence. The boundary entropy of c is given by $H_B(c) = -\sum_{h=1}^{m} P(h,c)log(P(h,c))$, where $P(h,c)$ is defined as $P(h,c) = Pr(s_{k+1} = e_h|s_j, \ldots, s_k)$. In other words, the boundary entropy of a sequence is the entropy of the subsequent token that follows the sequence, conditioned on the sequence. Said another way, it is the expected Shannon information of the token following the sequence.

VE uses a sliding window to make local splitting decisions. This allows the algorithm to run in linear time with respect to the size of the dataset, and therefore it can run on very long sequences. The *VE* algorithm consists of three main steps. Given a sequence of characters for segmentation:

Step 1

Build an n-gram trie of the sequence and use it to calculate the internal information and boundary entropy of each subsequence of length less than or equal to n. The trie is a simple tree structure that organizes and counts n-grams in an efficient way, such that they can be queried in O(log(n)) time. Each value is then standardized across all subsequences of the same length. Let H_I^l be the average internal information for all sequences of length l, and σ_I^l be the standard deviation of the internal information for all sequences of length l. Then the standardized internal information of a chunk c of length l is defined as $\hat{H}_I(c) = (H_I(c) - H_I^l)/\sigma_I^l$. The boundary entropy is standardized in exactly the same way, such that $\hat{H}_B(c) = (H_B(c) - H_B^l)/\sigma_B^l$.

Step 2

Pass a sliding window of length N along the sequence. At each location, let each of two experts vote on how they would split the contents of the window - one minimizing the internal entropy of the two induced subsequences, the other maximizing the entropy at the split. More precisely, given a window $w = (x_1, \ldots, x_N)$, expert 1 votes to break w into two

chunks $c_1 = (x_1, \ldots, x_i)$ and $c_2 = (x_{i+1}, \ldots, x_N)$ such that $\hat{H}_I(c_1) + \hat{H}_I(c_2)$ is minimized. Expert 2 votes to break w into two chunks $c_1 = (x_1, \ldots, x_j)$ and $c_2 = (x_{j+1}, \ldots, x_N)$ such that $\hat{H}_B(c_1)$ is maximized (see Figure 2.1). Both experts use the trie to perform these calculations.[1]

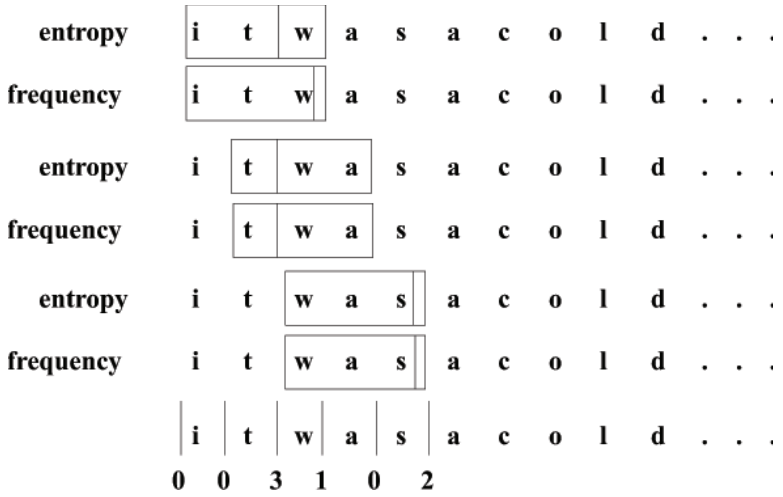

Figure 2.1 From Cohen et al. (2007). This figure illustrates the function of VE with $N = 3$ on the first few characters of George Orwell's "1984." At each step both experts vote how to break the contents of the window. In the original paper, Cohen used n-gram frequency to approximate internal information, which is why one expert is labeled "frequency."

Step 3

Choose a threshhold V_t. Induce a split at each point in the sequence that recieved more votes than its neighbors, so long as its total number of votes is greater than V_t. Essentially, this threshold is used to ignore peaks with a small number of votes (*e.g.*, a location that received 1 vote while its neighbors received 0). Lowering V_t causes the algorithm to induce more breaks,

[1]Instead of directly minimizing the internal entropy of induced subsequences, the original *VE* maximized the frequency, since the entropy of a sequence is given by the log of its frequency [Cohen et al. (2007)]. In our implementation of *VE* we did not maximize the frequency but instead minimized the entropy of the induced subsequences. This change caused a slight improvement in the baseline performance of *VE* of about 1%, which accounts for the difference between our results and those in the original paper.

and raising it leads to fewer. It does not change how the votes are cast, but simply how confident the algorithm must be at a break location.

For further technical and implementation details of the algorithm, discussion of the roles of N and V_t, or for a comparison of *VE* with other segmentation algorithms, see the journal article [Cohen et al. (2007)].

2.3 Hypothesis

The *VE* model bears a strong resemblance to the statistical learning approach mentioned before. If the conditional probability between each syllable within a word is high, then by definition the internal information of the word is low. But instead of evaluating each transitional probability in isolation, *VE* looks for short sequences of tokens where all of the TPs are high. Similarly, the boundary entropy of a sequence is high precisely when there is no particular token that is very likely to come next. However, instead of focusing on the transition probability between two syllables that happened to be adjacent, *VE* looks at whether the TP is *expected* to be high. This is an important difference, and it solves one of the major problems with the transitional probability approach. When the last syllable of one word and the first syllable of the next happen to form a likely pair, the TP based approach fails. But *VE* isn't affected when the TP at the word boundary is high, as long as the next token is unpredictable based on several previous tokens. This extra power is afforded by the use of the more sophisticated information metrics. Moreover, the model should still be extremely sensitive to the transitional probability cues, since the entropy cues must be present wherever the TP cues are.

I suggest that this is the solution to the previous conundrum regarding statistical learning. That is, that segmentation based on TPs works very poorly, and yet children seem to be able to segment sequences based solely on TPs. The solution is that children have a more powerful model of distributional segmentation, perhaps one akin to *VE*. Since statistical learning seems to be the first step toward language segmentation, this makes *VE* an attractive candidate to attempt to actually segment speech.

The central hypothesis of this thesis is that the *VE* model is a good model of the dis-

tributional aspects of the human audio segmentation mechanism, and that it can be used to segment audio data. This application is not trivial or straightforward, since the *VE* algorithm is designed to segment token sequences with a small alphabet. The following chapters will detail methods by which to apply *VE* to acoustic streams, and also ways to improve *VE* itself.

CHAPTER 3. ACOUSTIC SEGMENTATION EXPERIMENTS

This chapter[1] is divided into two main segmentation tasks. The first is an experiment which reproduces the original Saffran results described in Chapter 2. The second is an attempt to run the segmentation algorithm on a larger, natural language dataset. The work is organized to address and illustrate several key points. First of all, human infants are known to be able to segment acoustic stimulus streams based solely on distributional cues, particularly when those distributional cues are very salient. This is known precisely because of the work of Saffran et al. (1996). Any attempt to reproduce this ability must at least be able to reproduce these basic results. Secondly, for a distributional segmentation strategy to be useful, it must be able to find word breaks in natural language as well. The artificial language used in the infant experiments was specifically designed to be segmentable using simple statistical cues. Natural language does not exhibit such simple markers for word boundaries, and therefore presents a much more difficult challenge for distributional segmentation algorithms. *VE* has demonstrated the ability to segment the phonetic transcripts of natural speech [Miller and Stoytchev (2008c)], but whether this will extend to natural language audio streams remains to be seen.

As previously mentioned, the *VE* model is the chosen segmentation strategy in all of the following experiments. It has been noted that the indicators of low internal entropy and high boundary entropy are intuitive extensions of the transition probability strategy suggested by the statistical learning community. But it should also be noted that the statistics required to calculate those two quantities may be approaching the horizon of what can be heuristically approximated by the human brain. No one is suggesting that infants are taking the logs

[1]The work presented in this chapter appears in Miller and Stoytchev (2008b) and Miller et al. (2009).

of probabilities to determine likely segmentations of their world. However, it is easy to pay attention to the predictability inside of certain sequences, and the unpredictability at their edges. So it is reasonable to presume that humans might be sensitive to metrics similar to those in the *VE* model. It is, of course, unknown whether this is actually true, but it is certainly plausible.

3.1 Infant Experiments Repeated

We obtained two stimulus streams from the original infant speech segmentation experiments [Saffran et al. (1996)]. Each audio stream is 60 seconds long and contains roughly 90 "words." The first stream (stream A) was composed of randomly ordered instances of the four words *tupiro*, *golabu*, *bidaku* and *padoti*. The second stream (stream B) was composed of random instances of the words *tilado*, *dapiku*, *pagotu* and *burobi*. The second language is composed of the same syllables as the first, but arranged so that the concatenation of words in either language cannot produce a word from the other. So in some sense these two audio streams are disjoint.

In the original experiment, the infants were played a stream composed the same way as stream A, and then tested on a single word repeated over and over. This method is useful when evaluating infants because it is simple. However, a more thorough evaluation of the model can be performed, since it produces explicit break locations. It is more informative to test the model by training it on one stimulus stream and then testing it on the other. This provides more information on the performance of the model, but the results can clearly be compared to those of the infant experiments.

In order to evaluate the segmentations induced by the algorithm, we manually recorded the timestamps of all phoneme and word boundaries in the two stimulus streams. It is impossible for this process to be absolutely precise, since spoken audio is not actually composed of distinct phonemes. The sound morphs from one allophone to the next, providing few clear boundaries. However, the speech in the streams used by Saffran *et al.* is very regular, which allowed us to consistently place breaks at the same location in each word. The resulting "answer keys" were

consistent, but the true breaks in a word are, after a certain point, a matter of opinion. This is a fundamental problem in the evaluation of speech segmentation.

3.1.1 Acoustic Model

The *VE* algorithm as described in Chapter 2 is designed to segment a sequence of discrete tokens from a fairly small alphabet. In order to use this algorithm to segment acoustic speech, the audio stream must be represented as such a sequence. An acoustic model is needed - one that can represent an audio stream as a sequence of tokens. This is not trivial, and the choice of model will affect the segmentation performance of *VE*.

A raw audio clip C with n samples is merely a time series of n real values, *i.e.*, $C \in \Re^n$. An individual sample C_i represents the intensity of the pressure wave of the sound at time i. Another way to think of it is that C_i is the offset from neutral position of the speaker cone that is playing back the audio at time i. Audio is typically sampled at a very high rate in order to maintain the fidelity of the sound. In this thesis, all audio clips were recorded in a single channel at 16000 hertz. So a single second of audio was represented by a sequence of 16000 real values. This is hardly appropriate input for the Voting Experts algorithm.

Two main steps are required to change the raw audio signal into a time series that *VE* can segment. First, the audio must be turned into a sequence of meaningful features. The intensity of the pressure wave at a single time step is not very useful information. However, using some simple DSP we can obtain a much more useful and relevant representation of the data. The second step is to use these extracted audio features to build a model that can discretize the sound sequence into a token sequence, which is appropriate for segmentation by *VE*.

3.1.2 Audio Features

By far the most common features extracted from an audio sequence are the spectral features, which are calculated using the discrete Fourier transform. The transform must be performed over a short window of samples, usually numbering some power of 2. The Fourier transform essentially decomposes the waveform into a number of sine functions, solving for both their

amplitude and phase. The amplitudes of these sine waves indicate the intensity of the sound at different frequency levels over the given time window. This is essentially a decomposition of the sound into different frequency bins, and it is tremendously more informative for our purposes than the waveform itself.

In order to represent a long audio stream using spectral features, it's sufficient to break the stream into many small windows, and calculate the spectral features of each one. Typically the windows overlap by 1/3 to 1/2 of their width. In order to downplay the significance of the overlap, the values of the samples in each window are faded in and out at the edges. Most commonly a "Hamming window" is used. Given a sequence s of N real values, the Hamming window of s is a sequence h of N real values such that

$$h_i = w(i)s_i$$
$$w(i) = 0.54 - 0.46\cos(\tfrac{2\pi i}{N-1})$$

This essentially weights the samples in the middle of the window more strongly, so that the boundaries have less of an effect on the spectral features. Using overlapping Hamming windows and the discrete Fourier transform we can convert a waveform into a sequence of feature vectors, which represent the intensity of the sound in different frequency bins over time. This sequence is typically called the "spectrogram" of the audio stream, and can be visualized as in Figure 3.1.

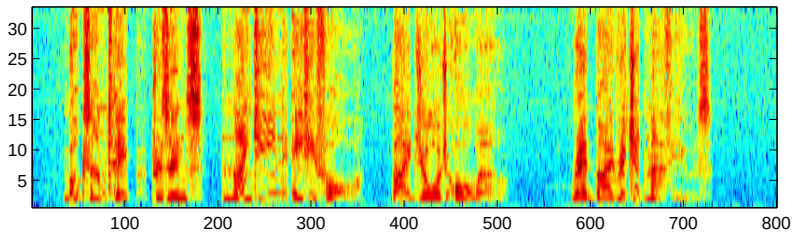

Figure 3.1 A spectrogram of the first few seconds of one of our audio datasets. The vertical axis represents frequency bins, and the horizontal axis represents time.

I used the raised cosine windower, the pre-emphasizer and the discrete Fourier transform in the Sphinx software package to obtain the spectrogram of each stimulus stream [Walker et al. (2004)]. The Fourier Transform was performed at 512 points. However, since we are only concerned with the power of the audio signal at each frequency level, and not the phase, the points are redundant. Only the first 257 points contain unique information. This transformation converted a 16kHz mono audio file into a sequence of spectral features, representing the intensity information in 257 frequency bins, taken every 10ms.

3.1.3 Vector Quantization

After converting an audio stream into a sequence of feature vectors, that sequence of vectors must be represented by a sequence of discrete tokens. The most immediate and straightforward strategy for creating such a representation is through vector quantization.

Vector quantization consists in representing a feature space using a small set of prototypical vectors, and then mapping each feature to a single one of those prototypes - typically the "closest" one. This is an extremely well studied problem, and there is a tremendous amount of literature devoted to it. Since innovation in the field of vector quantization is not a goal of this thesis, I chose to use an off-the-shelf software package and a standard, unsupervised algorithm to perform the task.

A Self-Organizing Map (SOM) [Kohonen (1988)] was trained on the feature vectors extracted from the audio. In order to avoid specifying the number of SOM nodes a priori, I used a Growing Grid SOM (GGSOM). A GGSOM is a self-organizing map that automatically grows to an appropriate size [Fritzke (1995)]. It adds nodes to the SOM until the variance of the instances mapped to any individual node is less than τ times the variance of the entire dataset, where τ is the "error parameter." This effectively ensures that no single node will account for more than τ of the total error. This way the SOM ends up sized appropriately for the particular problem, and the data is mapped roughly evenly among the nodes. For the experiments I chose a $\tau = 0.05$. I used the implementation of a Growing Hierarchical SOM (GH-SOM) in the Java SOM Toolbox to train our Growing Grid [Dittenbach et al. (2000)].

The GGSOM grew to contain 15 nodes for each stimulus stream. One would expect an SOM trained on spoken audio to have many more distinct states, but the stimulus streams are extremely limited, containing only 12 distinct syllables repeated over and over. Since the GGSOM automatically chooses the appropriate number of nodes, 15 must be enough to represent the data streams.

After training a GGSOM on the data it was used to classify each feature vector based on which node in the SOM it was most similar to (see Figure 3.2). This process is analogous to the technique of building a "codebook" that is often used for speech recognition [Rabiner (1990)]. Typically the k-means algorithm is used to cluster the time slices, but in this case the GGSOM was convenient since it automatically grew to an appropriate size.

In the resulting sequence, it was common for several consecutive instances to be mapped to the same node in the SOM. For instance, silence always maps to the same SOM node, so any period of silence in the original audio was represented by several instances of the same node in the discrete sequence. This also happened when a single sound was held for any length of time. In order to be time independent, these repeated sequences were collapsed into just one instance of the given node. This effectively denotes a prolonged period of the same sound by a single state (see Figure 3.2). This technique has also been successfully used to discretize natural sounds that were then used to classify objects based on their acoustic properties [Sinapov et al. (2009)].

3.1.4 Segmentation

The acoustic model then produced a sequence of node labels without any repeated states. Ideally, this sequence would correspond to the sequence of the most salient sounds in the acoustic stream. More importantly, the tokens of the sequence are discrete, and drawn from a reasonably sized alphabet. This is precisely the sort of sequence that *VE* is designed to segment. Accordingly, in order to segment the acoustic streams, I ran *VE* on the sequences generated by the acoustic model. It induced breaks in the sequence, and after accounting for the removed repeated states, those breaks could be mapped to timestamps in the original audio

stream (see Figure 3.2). This is possible since each spectral feature vector, and therefore each SOM node label, corresponds to a time window with constant width and overlap. The break was assumed to be directly in between the centers of the two adjacent time windows.

There currently exists no principled way to set *VE*'s parameters N and V_t. For the purposes of this experiment, I set $N = 7$ and $V_t = 2$. N was chosen to roughly approximate the expected length of an average word, and also to be consistent with previous work. In all of Cohen's experiments, N was also set to 7 [Cohen et al. (2007)]. The mean number of votes per location is always 2, due to the nature of the algorithm. Therefore, V_t was chosen such that a break could be induced if a location received more than the average number of votes.

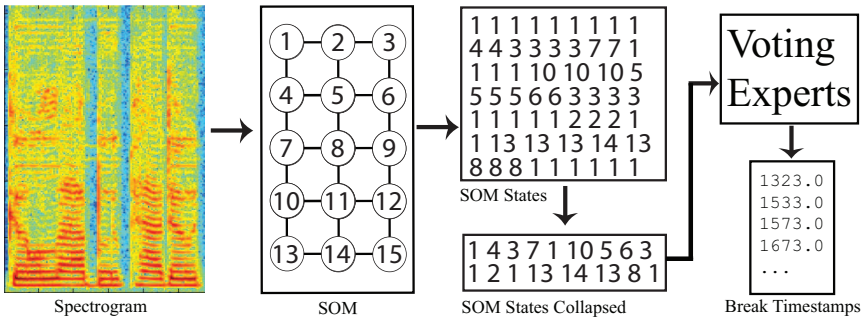

Figure 3.2 The audio segmentation process: The spectrogram of an audio stream is used to train an SOM, which is then used to label each time slice of the spectrogram. The repeated labels are removed, and that sequence is used to train the Voting Experts model, which then segments the sequence and specifies the timestamps of the induced breaks (in milliseconds).

This outlines a general, unsupervised algorithm for the segmentation of an audio stream. First, use an FFT to obtain the power spectrum of the audio stream. Then train a GGSOM on that data to cluster the time slices of the spectrogram into discrete tokens. Generate a new sequence of discrete tokens corresponding to the labels of the SOM nodes closest to each time slice in the spectrogram. Remove repeated labels. Run *VE* on the tokenized sequence and determine the timestamps of the induced breaks.

3.1.5 Evaluation Methodology

In order for an induced break to count as a correct break, it had to be placed within 13ms of a true break location. The reason the breaks were given a 13ms window on either side is that Sphinx uses a 26.6ms wide Hamming window to calculate the spectrogram information. The breaks produced by the algorithm correspond to the center of that window. An induced break was counted as "correct" if there was a true break anywhere inside that window.

A distinction was also made between breaks between phonemes and breaks between words. When marking the true breaks in each stimulus stream, the exact beginning and end of each word was recorded. Instead of marking a single break location between words, this specified a window in which the break must occur. The time between some pairs of words was trivially small. The time between others was longer. However, since the stimulus streams were generated artificially, the time between word pairs was consistent. An induced break was counted as breaking two words if it was placed anywhere in the window between them, plus or minus 13.3ms. This made the evaluation of word breaks much more reliable than the evaluation of phoneme breaks. The true break location between a pair of phonemes can be indeterminate. So it is sometimes illegitimate to specify a break and then expect a segmentation algorithm to induce that exact break within 13.3ms. However, the boundary between words can be more clearly delimited, and we can be certain that the true word break lies in the window specifying that boundary.

Unfortunately these large boundaries make it easier for the algorithm to accidentally induce a break between two words. Thus, even random breaks will be counted as correct a significant portion of the time. Accordingly, we used a Monte Carlo method to simulate random segmentations for each experiment. Each reported result is accompanied by the results of inducing 100 random segmentations, each one having the same number of induced breaks as the algorithm produced. These random trials are aggregated and provide a baseline from which to evaluate the algorithm.

Two metrics were used to evaluate the induced segmentation of each experiment. The first is the accuracy of the induced breaks. If t is the number of true breaks induced by the algorithm,

and n is the total number of breaks it induces, then the accuracy is given by $a = t/n$. The complimentary metric is the hit-rate. If m is the total number of true breaks in the stimulus stream and s is the number of true breaks that were also induced by the algorithm, then the hit rate is given by $h = s/m$. For each experiment the accuracy and hit-rate of the induced segmentation was computed over all breaks, including word breaks. Then the accuracy and hit-rate of the segmentation was also computed when only considering the word breaks. That is, the accuracy and hit-rate are reported as if the answer key contained only the word breaks. Word breaks are, in some sense, more important than phonemic boundaries, and this is why this additional evaluation was performed.

3.1.6 Experimental Results

The algorithm specified above constitutes a basic application of the *VE* model to a real audio stream. The first question is whether this can induce an accurate segmentation. The second question is whether we can use this system to model the human segmentation mechanism. The following experiments were designed to answer both of these questions.

Experiment 1: The segmentation process described above was run on both stimulus streams to obtain the induced breaks. Then the induced breaks were compared to the true breaks for each stimulus stream. The results are shown in Table 3.1.

The segmentation induced on both audio streams was significantly more accurate than chance. In particular, the algorithm found the vast majority of the word breaks in both cases. Note that the accuracy drops less than expected when evaluating on all breaks versus evaluating over just the word breaks. This means that most of the correctly induced breaks were at word boundaries. For example, stimulus stream A contains 265 true breaks, 89 of those being word breaks. The algorithm induced 205 breaks on the stream. Of those 205 breaks, 112 of them corresponded to true breaks. It correctly labeled 70 out of the 89 word breaks, but only 42 of the 176 phoneme breaks.

This is somewhat surprising, since there are twice as many breaks between phonemes as breaks between words in the sequence. However, as discussed earlier, the placement of the

Table 3.1 Results for Experiment 1.

Key	Stream	Accuracy (Rand)	Hit Rate (Rand)
All Breaks	A	0.546 (0.120)	0.411 (0.085)
	B	0.441 (0.131)	0.387 (0.107)
Word Breaks	A	0.341 (0.082)	0.764 (0.166)
	B	0.308 (0.091)	0.791 (0.214)

phoneme breaks in the answer key is much more subjective than the placement of the word breaks. Additionally, the information theoretic markers used by *VE* may be more consistently expressed at the word breaks. In any case, it is clear that this algorithm is adept at finding word boundaries in these stimulus streams.

Recall that these models were trained on only one minute of audio, containing roughly 90 spoken words. Even though the audio is simple and regular, this is still a very promising result, and definite proof that this model has the potential to segment streams of speech.

Experiment 2: The point of this experiment is to demonstrate that an SOM trained on stimulus stream A can still capture the information in stimulus stream B. The two streams are composed of the same set of syllables. The only difference is the order in which the syllables are heard, which may produce some interaction effects that the SOM cannot capture. However, most of the sounds are the same, so the tokenization of stream B based on an SOM trained on stream A should still be useful for inducing a tokenization on B.

To prove this, an SOM was trained on the spectrogram data of each stimulus stream to obtain SOM_A and SOM_B. Then SOM_A was used to tokenize the spectrogram data from stimulus stream B and vice versa. Then a *VE* model was trained on each of the token sequences and induced a segmentation. Once again the true breaks were used to evaluate the induced segmentation. The results are shown in Table 3.2.

There is a drop in both the accuracy and hit rate of each segmentation in this experiment. However, in each case the algorithm still performed much better than chance. Also, roughly half of the word breaks are still identified in both cases. While an SOM trained on stimulus stream A might not capture all of the sound information in stream B, it certainly captures

Table 3.2 Results for Experiment 2.

Key	Stream	Accuracy (Rand)	Hit Rate (Rand)
All Breaks	A	0.290 (0.126)	0.306 (0.127)
	B	0.279 (0.129)	0.244 (0.105)
Word Breaks	A	0.195 (0.086)	0.596 (0.250)
	B	0.184 (0.088)	0.473 (0.209)

enough to induce a decent segmentation.

Experiment 3: This experiment is intended to replicate Saffran's experiment on infants. In those experiments, the children listened to one stimulus stream, and were then presented a novel token. Similarly, in this experiment, our model is trained on one stimulus stream, and then used to segment the other. That is, the SOM and the statistical model of *VE* are learned from stream A, and then that model is used to segment stream B and vice versa.

Table 3.3 Results for Experiment 3.

Key	Stream	Accuracy (Rand)	Hit Rate (Rand)
All Breaks	A	0.167 (0.127)	0.011 (0.009)
	B	0.167 (0.163)	0.004 (0.004)
Word Breaks	A	0.000 (0.087)	0.000 (0.018)
	B	0.167 (0.123)	0.011 (0.008)

The algorithm is almost completely unable to induce a segmentation. In no case did it perform significantly better than chance. And, in fact, in some cases it performed worse. From the results of experiment 2 we can conclude that the poor performance is not the fault of the acoustic model. Instead, the language model trained on one language is insufficient to induce a segmentation in another.

The most interesting result was the number of breaks induced by the algorithm. Only 18 breaks were induced on stream A, and 6 on stream B. The reason for this lack of breaks is best illustrated by the votes cast by each expert. Recall that *VE* works by sliding a window along a token sequence, and using two experts to vote how to split the contents of the window at each location. The result is that at each possible break location (*i.e.*, between each pair

of tokens) some number of votes are cast, representing whether a break should be induced at that location. In theory, the true break locations should receive many votes, and the rest should receive few or none. Figure 3.3 shows the histograms of the number of votes cast at the break locations (*i.e.*, between the time slices) in stimulus stream A in both Experiment 1 and Experiment 3.

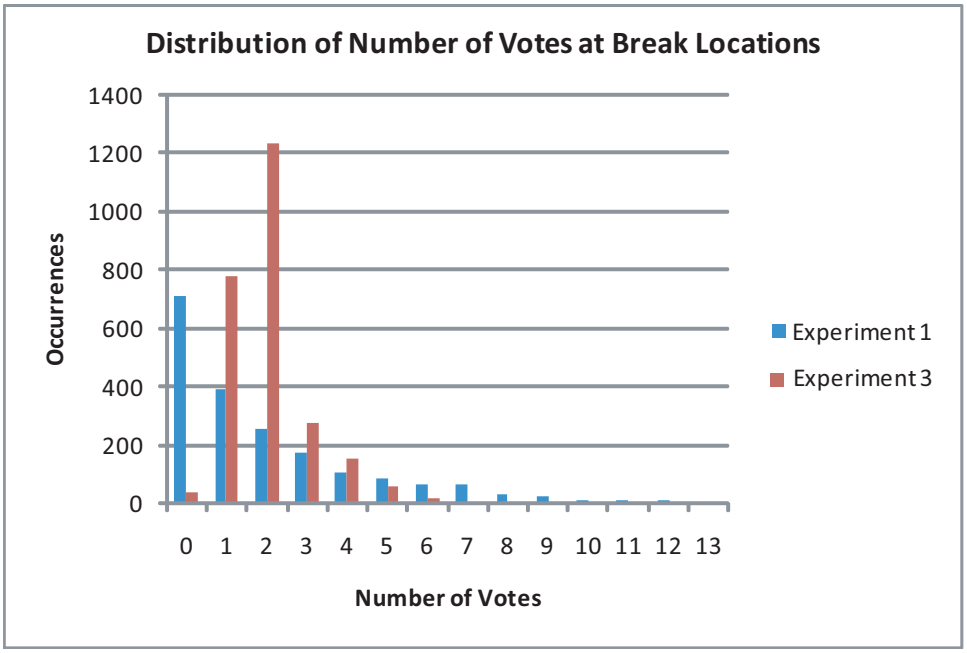

Figure 3.3 The histograms of the number of votes received at break locations for both Experiment 1 and Experiment 3. Notice the difference in vote distribution between Experiment 1 and Experiment 3.

Notice that the votes from Experiment 1 contain many zeros, and also many locations with a large number of votes. This means that the experts agreed on many vote locations, and voted for the same ones consistently. However, the votes from Experiment 3 display exactly the opposite characteristic. There are very few locations with 0 votes, and also very few with more than 4 votes. The model is "confused" by the data it is trying to segment. It lacks statistical information about the sound sequences and is therefore unable to distinguish

between common and uncommon audio chunks. It has little basis from which to calculate the internal or boundary entropy of subsequences. Therefore, the votes are spread more evenly among the break locations, they rarely exceed the threshold V_t, and so almost no breaks are induced. If V_t is lowered then more breaks are induced, however they are extremely inaccurate and do not improve the performance.

This corresponds precisely with the situation of the 8-month-old who listens to stimulus stream A, and then hears a novel word. The child has learned the sounds present in the stream, and has learned a statistical model that characterizes it. Then, suddenly, that model is violated. The child is initially unable to use the old model to "understand" the novel word, and therefore becomes confused.

3.1.7 Summary

This section shows that the *VE* model is capable of inducing an accurate segmentation on an audio stimulus stream with very limited training data. It also shows that the behavior of this model mimics the behavior of 8-month-old infants. This should be counted as a small victory for both the hypothesis of statistical learning and the *VE* model. It is possible to use statistical information theoretic metrics to automatically induce word boundaries in an audio stream. Specifically, the low internal entropy and high boundary entropy of chunks provide sufficient markers to do so. Additionally, the algorithm has passed the requisite test for any suggested model of human segmentation - it has behaved in the same way as the infants. Had the model behaved differently it would have precluded it from plausibility. But what is still unknown is exactly how effective such a model might be on more complex human speech. The next section addresses this question directly.

3.2 Segmentation of an Audio Book

In order to further evaluate the segmentation algorithm specified above, it was used to segment the audio from the first of 9 CDs taken from an audio recording of George Orwell's novel "1984." The audio file was roughly 40 minutes in length. The reason this particular audio

book was chosen is that the text from 1984 was used in the original segmentation experiments with VE [Cohen et al. (2007)]. This experiment was performed to evaluate the procedure's effectiveness on language spoken by a person, as compared to artificially generated language.

3.2.1 Evaluation Methodology

Evaluating the output of the algorithm proved to be very difficult. The segmentation was tested by using human volunteers to verify the breaks. For each induced break, they checked the audio stream to see whether it was placed correctly. In order for it to count as a correct break, it had to be placed within 13ms of an obvious break location. Such locations include the beginning and ending of words, as well as phoneme boundaries where the audio stream suddenly changes in intensity or frequency. Any break placed in the silence between words or phonemes was also counted as correct. These locations were verified visually using software we wrote to view the waveforms and the breaks.

Unfortunately, it was impossible to measure the hit-rate of the induced breaks in this case, since the number of true breaks in the stream was not known. The segmentation could only be evaluated in terms of the accuracy of the induced breaks. However, the accuracy of the induced breaks is not sufficient for a complete evaluation. After all, for a break to be considered "correct" it simply had to fall within 13ms of any artifact in the audio stream that a human would consider a logical breaking point. As it turns out, this is actually pretty likely. To account for this, the performance of our algorithm was measured against randomly generated breaks.

Over the 40 minutes of audio, the algorithm induced roughly 16,000 breaks. It would have been impossible to manually check every single one of them. Instead, random subsets of the data were chosen for evaluation. Five sections of 1 minute each were randomly chosen from the audio stream, each one containing roughly 400 breaks. The breaks in these 1 minute segments were recorded. For each of these sections the same number of break timestamps over the same 1 minute were randomly generated. We wrote a Java program to load all the breaks at once, visualize the waveform, and allow the volunteers to scroll through the breaks and make their

decisions quickly (see Figures 3.4 and 3.5).

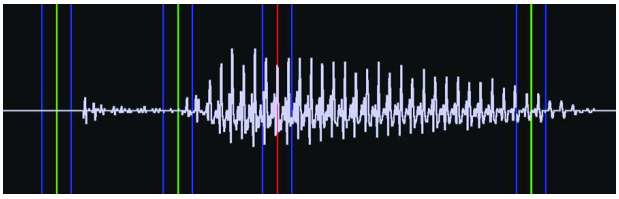

Figure 3.4 A short sample of the audio from the experiment which shows the breaks generated by our algorithm. The "correct" breaks are highlighted in green. The two blue lines around each break show the bounds of the 26ms window.

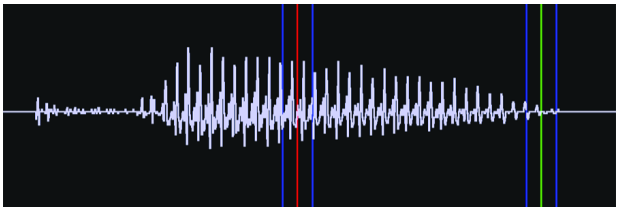

Figure 3.5 The same audio segment as shown in Figure 3.4, but with the randomly generated breaks shown instead.

There are several reasons to think that "spot checking" the algorithm is a good measure of its overall accuracy. The time windows used to check the segmentation were chosen randomly, and therefore should be representative of its overall quality. Also, enough points were sampled to ensure that the estimated accuracy was reliable.

The volunteers were not told which file contained random breaks, and were instructed to use their best judgment to determine the correctness of each break. They were specifically told to use consistent judging criteria between the two files. This way the performance of this algorithm was compared to an algorithm that randomly generates roughly the same number of breaks. Figure 3.4 shows an sample section of audio with the induced breaks drawn in. Figure 3.5 shows the same section of audio with the randomly generated breaks. These sections have already been graded, and the "correct" breaks are marked in green.

3.2.2 Intercoder Reliability

Two volunteers were trained how to use the visualization software, and how to visually identify breaks in the audio stream. One grader was chosen to grade all 10 break files (5 generated by the algorithm and 5 generated randomly), to maintain consistency over the entire dataset.

The second grader was used to test intercoder reliability (Holsti 1969). That is, how consistently human beings will make the same grading decisions regarding the correctness of an induced audio break. The second grader was trained separately from the first, and given 2 pairs of files to evaluate. Thus, the second grader evaluated roughly 40% of the same data as the first grader. If t is the total number of decisions to be made by two graders and a is the number of times they agree, then the intercoder reliability is given by $ICR = a/t$. The ICR values for both experiments are summarized in Table 3.4. The agreement between our graders is fairly high, considering the subjective nature of most audio break judgments. Typically, an intercoder reliability of 0.8 is considered acceptable.

Table 3.4 The Intercoder Reliability

Total Breaks	Agreed Breaks	ICR
1564	1356	0.867

3.2.3 Results

The graded segmentation results are shown in Table 3.5. The breaks induced by the algorithm are shown next to the breaks that were randomly assigned. As you can see, the *VE* segmentation performed substantially better than chance. The accuracy was above 80%, which is considerably high. However, the probability of a random break being placed at a correct location is above 60%. It is impossible to know whether a significant portion of the algorithm's breaks were placed "randomly," and then accidentally marked as correct by the graders.

However, anecdotally, the breaks induced by *VE* were much more logical than the random ones, even in cases when its breaks were incorrect. They tended to be placed at the beginning

Table 3.5 Accuracy Results

	Total Breaks	Correct Breaks	Accuracy
Algorithm	1910	1538	0.805
Random	1910	1220	0.639

and ending of words, and at dramatic shifts in the audio signal. Many times the "incorrect" breaks came at locations that could have been phoneme boundaries, but were impossible to distinguish visually by the graders. An audio evaluation of each break would have taken a substantial amount of time compared to the quick visual grading, and we lacked the manpower and resources to perform that experiment.

Also, the randomly generated breaks got a substantial number "correct" that happened to fall in the silence between words. We instructed the volunteers to count any breaks that fell in silence as correct, so these breaks helped to increase the accuracy of the random segmentation. The VE breaks, however, generally did not exhibit this behavior. They were usually placed either neatly between the words, or at the end of the silence before the next word began. These qualitative observations are not reflected in the difference in accuracy between the VE segmentation and the random one. Which leads us to believe that further evaluation will show a much greater gap between the two methods. Figures 3.4 and 3.5 illustrate a typical example of the difference in segmentation.

3.2.4 Summary

This experiment demonstrates the acoustic segmentation algorithm's ability to work on real world audio, as well as its tractability when dealing with large datasets. The segmentation and evaluation, however, are imperfect. This is an unfortunate consequence of working with such a large, unlabeled natural language stream. However, despite these limitations, the algorithm's performance was significantly better than chance. Therefore, the distributional cues used by *VE* must be useful for the unsupervised segmentation of natural language. Not only can this model reproduce the behavior of 8-month-old infants using artificially generated speech, but it

can also identify breaks in more complex spoken audio. The next logical question is whether the use of a more sophisticated acoustic model might improve segmentation performance, and this is the subject of the next chapter.

CHAPTER 4. AN IMPROVED ACOUSTIC MODEL

The previous chapter showed that *VE* can be used to segment audio streams. This chapter[1] introduces a more complicated acoustic model that's similar to more traditional speech recognition systems. This model is then used to repeat the infant experiments from Chapter 3. The purpose of this experiment is to demonstrate that an improvement in the acoustic model translates directly into an improvement in the performance of the segmentation algorithm. This represents a step toward an unsupervised acoustic model that is robust and powerful enough to represent human speech well enough for a distributional segmentation algorithm to break it into words.

4.1 Improved Acoustic Model

The acoustic model described in Chapter 3 used the Fourier transform to extract the spectral features of a given audio stream, and then quantized those features to produce a state sequence. This is a simple, and older method that is rarely used in modern speech recognition systems [Rabiner (1990)]. Both the feature vectors and the model itself can be improved.

4.1.1 Mel-Frequency Cepstral Coefficients

The spectral features of an audio stream are certainly useful, but may not be the most ideal for dealing with human speech. In fact, the most common features used for speech recognition are the Mel-frequency cepstral coefficients and their first and second order time derivatives [Davis and Mermelstein (1980)]. These are calculated using the spectral features, and are based on empirical study of the sensitivity of human coclea.

[1]The work presented in this chapter has not yet been published, but it has been submitted for review.

In order to calculate the Mel cepstrum, the frequency intensities obtained from the spectral features are first passed through a filter bank. The bank typically consists of 40 different triangular filters, each one sensitive to a different frequency range. This step essentially smooths the power spectrum, and reduces its dimensionality down to just 40 values. The filters in the bank have been chosen to mimic the sensitivities of the human ear. That is, they are spread through the acoustic spectrum in roughly the same distribution to which humans are responsive.

The values obtained from the filter bank are then passed through a cosine transform, to obtain a set of 13 cepstral coefficients. So, essentially, the cepstral features are a transform of a transform. They represent oscillations across frequency bins, which are very common in human speech. When our vocal chords make a tone, they also emit sound at several harmonic levels above the fundamental frequency. Additionally, the first coefficient accounts for the intensity of the sound across the entire sample. The remaining 12 coefficients give a volume-independent representation of the sound. Along with these 13 coefficients, it's common to include their first and second time derivative as features. This is calculated by fitting a line to the points in a small time window, usually extending back 5-9 samples. Also, the value of the first coefficient increases exponentially as the perceived volume of the sound increases linearly, so often the log energy of the signal and the log energy of each derivative are also included. Together, the Mel frequency cepstral features are composed of the 13 cepstrum, 13 first order and 13 second order derivatives, and the log energy of each. This gives a grand total of 42 features per sample. In all experiments described in this chapter, the acoustic streams were first converted into a sequence of Mel-cepstrum feature vectors in precisely this way. This is a standard method of feature extraction for speech processing, and it was performed using the Matlab package "Voicebox."

4.1.2 Modern Speech Recognizers

The design of modern speech recognition systems is extremely complex. Most often they use a hierarchy of Markov models and rule based systems for grammar and syntax. Most of

these components are not applicable to the task at hand. However, we can focus specifically on typical phoneme recognition models. These models sit at the lowest level of the speech recognizer, and attempt to label the sequence of acoustic feature vectors as a sequence of phonemes. I will describe a stereotypical phoneme model, and then demonstrate how a similar model can be used to improve the quantization of our acoustic feature vectors.

Typically each phoneme is represented by it's own Gaussian Mixture Hidden Markov Model (GMHMM). There are several common graph architectures that are used for these phoneme models, but the most common are the 3-node and 4-node skip chains. In this thesis, we always use a 3-node Markov chain with Bakis topology for the phoneme model [Rabiner (1990)], so we will use that structure in all further examples (see Figure 4.1).

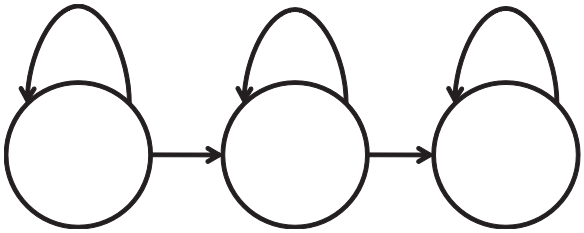

Figure 4.1 A 3-node Markov chain with standard Bakis topology.

In a discrete HMM each node in the graph defines a probability mass function (pmf) over the possible, discrete observations. In this case, the observation sequences are continuous real valued vectors with high dimensionality. So instead of a pmf, each node in the HMM represents the probability density of observations using a mixture of Gaussian functions. This completely specifies the likelihood of an observation given a state. We can then use standard HMM algorithms to calculate the most likely state sequence given a sequence of observation vectors, or calculate the likelihood of a sequence given the model. However, learning the model parameters is a bit more complex.

The means, covariance matrices, and mixture matrices of the Gaussian mixture models associated with each state must be learned through expectation maximization (EM). They must

be initialized, either randomly or through some other means, and then iteratively improved to maximize the likelihood of the data. Suppose we wished to train a 3-node Markov chain to recognize the phoneme "th," where each node was represented by a 1-Gaussian mixture model. This is the simplest case, where each node is not actually a "mixture," but rather a single Gaussian.

We might first use a phonetically transcribed and labeled audio speech corpus to obtain several hundred different pronunciations of the syllable "th." Each short audio clip would be converted into a sequence of Mel ceptral features, along with their time derivatives and the log energy. With 3 nodes and 1 Gaussian per node, we must initialize 3 means. One common method is to perform k-means clustering on all of the feature vectors from all of the clips with 3 means. Given these 3 clusters, we can estimate the 3 means, along with their covariance matrices, based on the feature vectors assigned to each cluster. These parameters can then be assigned to the three nodes of the Markov model.

Once the model is initialized, all that remains is to use EM to optimize the parameters. The maximization step consists of using Viterbi decoding to calculate the most likely state sequence given the observation sequence. Then the expectation step involves re-estimating the Gaussian mixture models given the feature vectors that are assigned to each state.

Typically this process, or one very similar, is repeated for each phoneme in the target language. Notice that this is a supervised process, and can only be accomplished using a large set of labeled spoken phonemes. The result is a set of roughly 40 Markov chains (depending on the language), each one trained to recognize a different phoneme. Their parameters are then improved by using EM to bootstrap over a large audio corpus.

We will draw inspiration from these models, however we cannot apply the techniques exactly. In the infant experiments the children learned to segment novel language streams in a completely unsupervised way. Therefore, any model of this process must also be entirely unsupervised. These HMMs are typically trained on labeled data, disqualifying them as plausible models. Specifically, a separate Markov chain is typically trained to represent each phoneme in the language. The models are built using a large set of hand-labeled instances of each

phoneme. Instead, we will suggest an unsupervised model that can convert an audio stream into a state sequence suitable for segmentation, but one that does not necessarily correspond to the phoneme sequence as a human would label it.

4.1.3 Unsupervised Acoustic Model

The critical observation is that we don't necessarily need a sequence that corresponds to the true phonemes of the language. All that's needed is a model that decomposes an audio stream into a sequence of its most salient acoustic features. These may or may not correspond to the "phonemes" as a human might label them. But that is irrelevant, at least as far as *VE* is concerned.

Just such a model was suggested by Iwahashi (2006), and implemented by Brandl et al. (2008). A version of that model is used in this work. Each phoneme was represented using a 3-node Markov chain with Bakis-topology, with the observation probability density of each state represented by a mixture of Gaussian functions [Rabiner (1990)]. In order to train these models without labeled data, a completely connected Markov network containing 10 Gaussian mixture states was trained on the acoustic stream. The parameters of the network were initialized using k-means, and then optimized using EM, so no labeled data was required. Then, paths of length 3 through that network were stochastically sampled based on the learned transition probabilities. Self-connections were ignored in the sampling, to ensure that the paths were composed of 3 unique states. The m most common paths were used to initialize m 3-node Markov chains. The last state of each chain was connected to the first state of every other chain, including itself, initialized with uniform transition probability (see Figure 4.2). The parameters of this larger Markov model were then optimized over the corpus using EM.

In one implementation, m was set using the Akaike information criterion [Brandl et al. (2008); Akaike (1974)]. Instead I used $m = 10$ to build the models used in this chapter. I did vary this parameter, and found that it did not have a strong effect on the performance of the model on these tasks. A thorough evaluation of its effect was not performed, since it

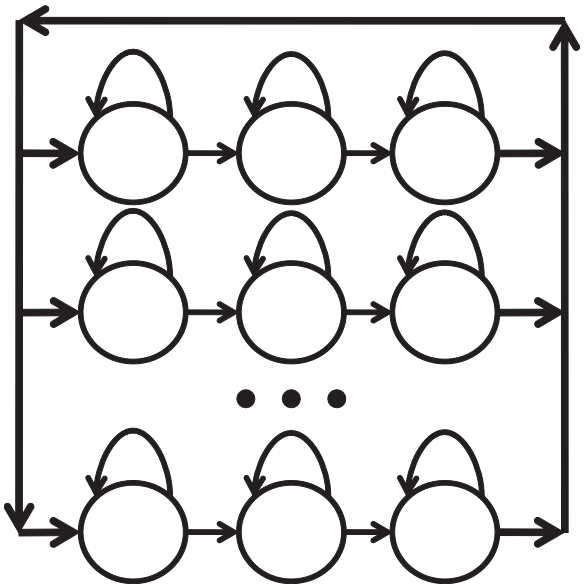

Figure 4.2 The structure of the entire Markov model after the individual chains are connected. Ideally, each chain should represent a unique phoneme in the language. The experiments described in this chapter use ten 3-node Markov chains.

was not considered germane to this investigation. However such an evaluation would certainly be worthwhile if this model were to be applied to other, more common speech recognition or learning tasks. It is sufficient, for our purposes, that this parameter could be set automatically. In other words, human infants might learn how many phonemes are in the language they hear, and learn to recognize each one.

4.1.4 Segmentation

Given a model as described above and an acoustic stream for segmentation, the stream was converted into a state sequence using Viterbi decoding. The state sequence was simplified by assuming that all nodes from the same Markov chain were equivalent. So instead of a sequence of nodes in the HMM, the stream was represented as a sequence of 3-node Markov chain labels. This created sequences with long stretches of the same label repeated over and over. These repeated labels were collapsed into a single token. So the final token sequence represented the order in which these chains were visited in the decoding of the stimulus stream, with no

information about how long the sound stayed in the same chain. If the chains corresponded to the phonemes of the language, as they do in more typical acoustic models, the result would be a transcription of the spoken phonemes of the stream. The idea is that the unsupervised model approximates the phoneme sequence, but perhaps extracts a slightly different set of fundamental sounds.

In order to segment the stream, *VE* was run on the resulting label sequence. *VE* placed breaks at locations of low internal entropy and high boundary entropy. Then, after accounting for the collapsed (*i.e.*, repeated) states, it produced the time stamps of all of the induced break locations in the audio stream. These timestamps were then used to evaluate the segmentation.

4.2 Infant Experiments Repeated

In order to evaluate the effect of the improved acoustic model, the infant segmentation experiments described in Chapter 3 were repeated. The segmentation of the audio book was not repeated, since its evaluation proved to be so difficult, and the results would have been somewhat uninformative.

4.2.1 Datasets

We used the same two stimulus streams that were used before, except, in this case, phoneme breaks were ignored. After performing the first set of experiments, it was concluded that the marked phoneme breaks were simply too unreliable to provide a decent evaluation of the segmentation. Furthermore, the goal is not to segment acoustic streams into phonemes, but instead into words. Therefore the breaks of interest are those that fall between word boundaries. Counting breaks between phonemes simply served to raise the measured "accuracy" of the induced segmentations by providing more "correct" locations. However, it is unclear whether those break locations should be counted as "correct" at all. For these reasons, they were excluded from this evaluation. In order to compare the two models, the results reported here should be compared with the results on "word breaks only" from Chapter 3.

4.3 Evaluation Methodology

In order for an induced break to count as a correct break, it had to be placed between the specified end of the previous word and the beginning of the next one, within an error of one time slice. The feature vectors that composed the audio stream were calculated using a window that was 0.016 seconds wide with a 50% overlap. This means that the additional time slice allowed at each boundary increased the break window by 0.008 seconds. This leeway was provided to compensate for labeling errors or other boundary conditions. Note that this is a tighter bound than the 13.3ms used in Chapter 3. The difference comes from the size of the window used to compute the Mel-cepstral feature vectors.

An induced break was counted as breaking two words if it was placed anywhere in the window between them. Both stimulus streams were 61.2 seconds long. Stimulus stream A contained approximately 7.7 seconds of "break" time, and stream B contained 7.2 seconds. The reason for the discrepancy is that the different pronunciations of the first and last syllables of the words in each stream led to slightly different amounts of time between them. It should be noted that these "breaks" are not perceivable when listening to the stream, and are no longer than the space between the phonemes within words (See Figure 4.3).

Unfortunately these boundaries make it easier for the algorithm to accidentally induce a break between two words. Thus, even random breaks will be counted as correct some of the time. Accordingly, a Monte Carlo method was used to simulate random segmentations for each experiment. Each reported result is accompanied by the results of inducing a large number of random segmentations, each one having the same number of induced breaks as the algorithm produced. The random breaks were induced in the same compressed state sequence used by *VE*, and were evaluated in the same manner. These random trials are averaged and provide a baseline from which to evaluate the algorithm.

The quality of the segmentation is evaluated based on the accuracy, hit-rate and f-measure of the induced breaks. In this case, accuracy is the percentage of induced breaks that are

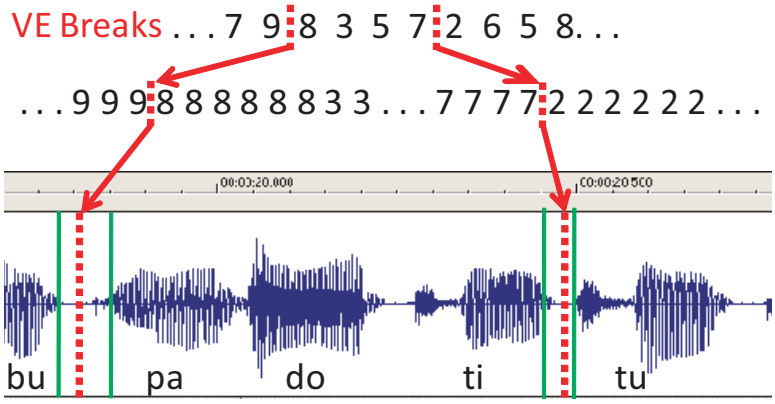

Figure 4.3 Evaluation of the breaks induced by *VE*. Each break is mapped to its location in the expanded state sequence, which corresponds to a timestamp in the audio stream. The break counts as correct if it falls within the marked boundary between two words. The states are represented by their numeric index in the Markov model.

correct, hit-rate is the percentage of true breaks found by the algorithm, and the f-measure is the harmonic mean of the two, given by

$$\text{f-measure} = \frac{2 * accuracy * hitrate}{accuracy + hitrate}$$

The f-measure is treated as most important, since it strikes a balance between the other two. It's possible to increase the accuracy of the segmentation by inducing fewer breaks, but being more confident about those that are induced. However, this will lower the hit-rate. Similarly we can raise the hit-rate by inducing more breaks, but this will lower the accuracy. The Voting Experts algorithm lets us explicitly make this trade off by adjusting the threshold V_t for the minimum number of votes required to induce a break at a location. All three of these metrics will be reported for each of the experiments. Additionally, the experiments will be repeated for a range of thresholds V_t, and the sensitivity of these metrics to variation in that threshold will be demonstrated.

It should be noted that the initialization of the acoustic models is a stochastic process, and leads to a unique model every time. The EM algorithm does not necessarily find a global optimum for the model parameters, but only a local maximum. Therefore, the model

should not be evaluated based on a single instantiation, but rather based on several trials. Accordingly, 10 different acoustic models were trained on each of the two stimulus streams. All three experiments were performed 10 different times with 10 different pairs of models. The results were averaged to produce the results reported. This is much more thorough evaluation of the model than the one performed in Chapter 3, since only one GGSOM was trained in that case.

Additionally, the segmentation step, where VE was run on the token sequence, was repeated for different threshold values V_t ranging from 1 to 8 for each experiment. Notice the trade off between accuracy and hit-rate as V_t varies. The f-measure, accuracy and hit-rate are reported both for the aggregate over all 10 models, as well as for the random trials over the same data. For each trial that was done with a single model, 10 random trials were performed. So, overall, 100 random trials were performed in each experiment for each stimulus stream.

4.3.1 Experimental Results

The procedure described above replaces the vector quantization method described in Chapter 3. Instead of training a GGSOM based on the spectral features of an audio stream, a series of 3-node GMHMM chains is trained on the Mel cepstral features. Instead of a state sequence corresponding to SOM nodes, the model produces a state sequence corresponding to chain labels. The output of both models has the same structure - a sequence of discrete tokens with the repeats removed. This sequence can be segmented and evaluated in precisely the same way as in Chapter 3. The hypothesis, however, is that the more complex model will more accurately represent the underlying acoustic stream, and the segmentation induced using it will be of higher quality. In order to test this hypothesis, the three experiments that were run on the infant dataset were re-run using the new model.

Experiment 1: The segmentation process described above was run on each stimulus stream individually. Then the induced breaks were compared to the true breaks for each stimulus stream. The results are shown in Figure 4.4.

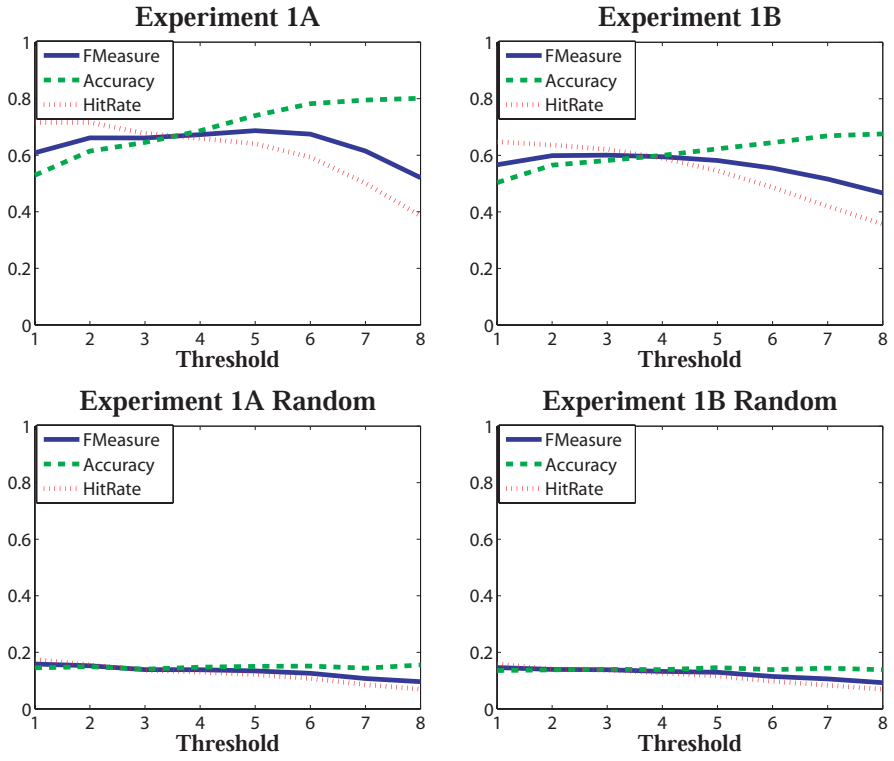

Figure 4.4 The F-measure, accuracy and hit-rate of the segmentation of both stimulus streams in Experiment 1, along with the performance of random segmentations on both datasets.

The segmentation induced on both audio streams was significantly more accurate than chance, and superior to the performance reported in Chapter 3. Table 4.1 shows a direct comparison between the performance of the GGSOM model and the more sophisticated GMHMM model. The values reported for the GMHMM model are those obtained when $V_t = 2$, which is the same threshold that was used for all experiments in Chapter 3. Notice that the hit-rate on word breaks reported in the previous chapter is higher than the hit rate in this trial. However, the accuracy is substantially lower. This means that the previous results correspond to a threshold that induces many more breaks in the streams, which actually serve to degrade its overall performance. As a result, the F-measures obtained in this experiment are substantially higher than those previously achieved.

Table 4.1 Comparison of results for Experiment 1 between the GGSOM used in Chapter 3 and the GMHMM described here. The comparison is done using the same value for V_t, which is 2.

Model	Stream	Accuracy	Hit Rate	F-measure
GGSOM	A	0.341	0.764	0.472
	B	0.308	0.791	0.443
GMHMM	A	0.615	0.716	0.661
	B	0.565	0.637	0.599

Experiment 2: The point of this experiment is to demonstrate that an acoustic model trained on stimulus stream A can still be used to segment the audio from stream B, and vice versa. The two streams are composed of the same set of syllables. The only difference is the order in which the syllables are spoken, which may produce some interaction effects that the GMHMM cannot model. However, most of the sounds are the same. So, for instance, the tokenization of stream B by an acoustic model trained on stream A should still be useful for inducing a segmentation on B.

Table 4.2 Comparison of results for Experiment 2 between the GGSOM used in Chapter 3 and the GMHMM described here. The comparison is done using the same value for V_t, which is 2.

Model	Stream	Accuracy	Hit Rate	F-measure
GGSOM	A	0.195	0.596	0.294
	B	0.184	0.473	0.265
GMHMM	A	0.381	0.545	0.449
	B	0.445	0.583	0.505

To demonstrate this, an acoustic model was trained on each stream to obtain GMHMM_A and GMHMM_B. Then GMHMM_A was used to tokenize the feature vectors from stimulus stream B and GMHMM_B to tokenize stream A. Then a *VE* model was trained on each of the token sequences and induced a segmentation. Once again the true breaks were used to evaluate

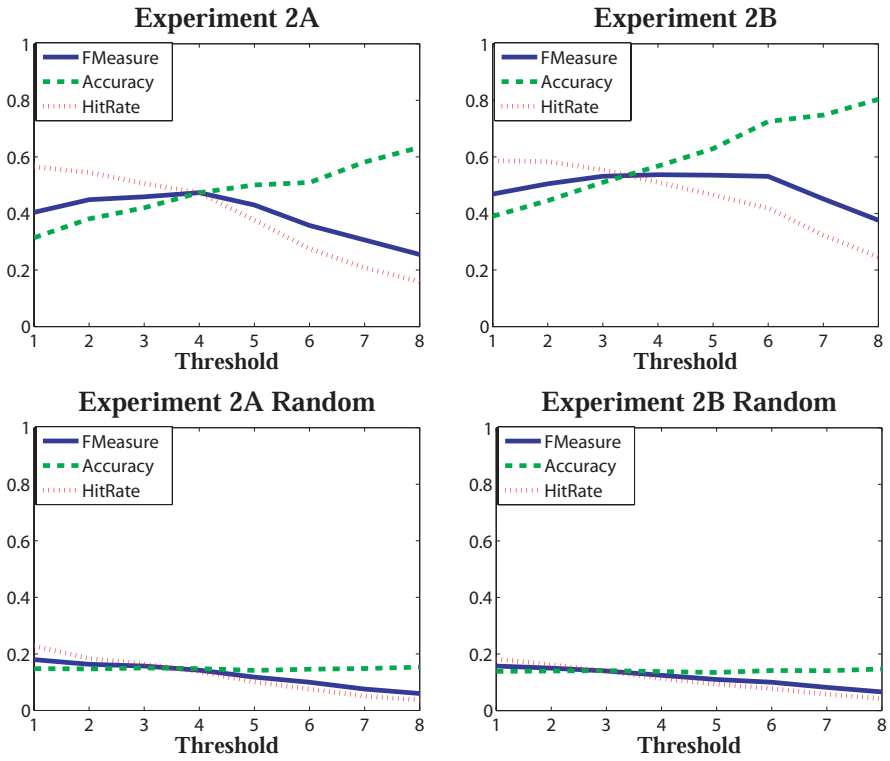

Figure 4.5 The F-measure, accuracy and hit-rate of the segmentation of both stimulus streams in Experiment 2. Once again, the performance of random segmentation is also shown.

the results (see Figure 4.5).

There is a slight drop in both the accuracy and hit rate of each segmentation in this experiment. However, in each case the algorithm still performed much better than chance. There is not a tremendous loss due to the unmodeled interaction of the diphones in the stimulus streams. Furthermore, the results presented here are far superior to those from experiment 2 in Chapter 3. Once again, the hit-rates from before were slightly higher than those shown here, but the accuracy was very much lower. The results are compared in Table 4.2.

Experiment 3: This experiment is intended to replicate the results of the infant studies. In those experiments, the children listened to one stimulus stream, and were then presented a novel token from the second stream. Similarly, in this experiment, the model is trained on one stimulus stream, and then used to segment the other. That is, the GMHMM and the statistical model of *VE* (the experts) are trained on stream A, and then that model is used to segment stream B and vice versa.

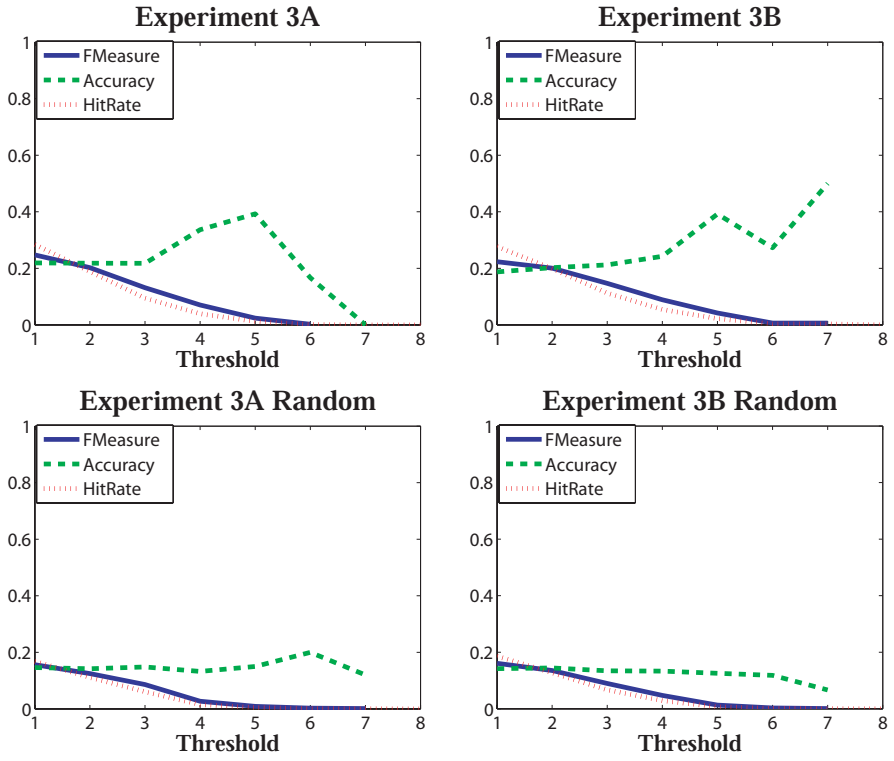

Figure 4.6 The F-measure, accuracy and hit-rate of the segmentation of both stimulus streams in Experiment 3, along with the results of the random segmentation.

The results show that the algorithm is almost completely unable to induce a segmentation. It performs only slightly better than chance, and this is most likely due to its ability to pick out syllables. From the results of Experiment 2 we can conclude that the poor performance is not the fault of the acoustic model. Instead, the language model trained on one language is insufficient to induce a segmentation in another.

As the threshold increases, the algorithm induces very few breaks. When V_t is higher than 5, almost no breaks are induced (*e.g.*, no breaks were induced at all when $V_t = 8$). This explains why the accuracy becomes erratic at higher threshold levels, and the hit-rate drops very low. The random segmentations only contained as many breaks as the algorithm induced, so the random hit-rate drops as well. The fact that not very many breaks were induced indicates that the experts did not vote for the same break locations very often. They could not agree on suitable breaking points, and therefore did not create many breaks.

This corresponds precisely with the situation of the 8-month-old who listens to stimulus stream A, and then hears a novel word from stream B. The child has learned the sounds present in the stream, and has learned a statistical model that characterizes it. Then, suddenly, that model is violated. The child is initially unable to use the old model to "understand" the novel word, and therefore becomes confused.

4.4 Summary

The results from these experiments precisely mirror those originally demonstrated in Chapter 3. The only difference is that the superior acoustic model utilized here lead to a much higher quality segmentation. Furthermore, more in-depth analysis of the role of V_t, the averaging over multiple acoustic models, the exclusive use of word breaks, and the focus on the combination of accuracy and hit-rate serve to make this evaluation much more reliable and informative than the previous results.

The most important conclusion to be drawn from these experiments is that a better acoustic model enables a distributional segmentation algorithm to obtain better results. In other words, one way to improve segmentation performance is to improve the quality of the representation of the acoustic stream. This is one possible direction of future work - developing more sophisticated unsupervised acoustic models.

This chapter has focused on the implementation and evaluation of an acoustic model, used to transform an audio stream into a state sequence suitable for segmentation using the *VE* algorithm. The next logical step is to consider whether the *VE* algorithm can be improved or extended itself. That is the focus of the next chapter.

CHAPTER 5. HIERARCHICAL VOTING EXPERTS

In the previous chapters the *VE* model has been demonstrated to be capable of segmenting acoustic language streams given an appropriate representation. However, those results relied upon the original *VE* as described in Chapter 2. As has been mentioned, the sliding window implementation is a heuristic that approximates the optimal segmentation based on the *VE* model. This chapter[1], suggests improvements and extensions of the original *VE* algorithm. Specifically, it will demonstrate how *VE* can be applied iteratively to a sequence, to produce a hierarchical segmentation. It will also suggest a strategy for adding a third voting expert that uses information from higher order models to improve lower level segmentation accuracy. Several experiments are performed to determine the behavior and effectiveness of these extensions.

5.1 Hierarchical Segmentation

Real world data often exhibits an inherently hierarchical structure, and it is well known that humans chunk the world hierarchically [Miller (1956); Fiser and Aslin (2005)]. When we read text our eyes scan the letters and sense black and white shapes. These shapes are chunked into letters, which are chunked together into words, which are chunked into phrases and so on. This hierarchical grouping is fundamental to our interaction with the world.

This chapter extends the *VE* algorithm to segment hierarchically structured sequences. It shows that *VE* can be generalized to work on hierarchical data and investigates the applicability of this extension to determine its strengths and limitations. More specifically, we will strive to understand when the underlying information theoretic model for segmentation is valid, and

[1]The work presented in this chapter won a best paper award at the IEEE International Conference on Development and Learning (ICDL) in 2008 [Miller and Stoytchev (2008a)]

when it is not. This chapter will also show how the higher order models can be used to improve the accuracy of the segmentation at lower levels.

5.2 Related Work

In addition to the segmentation algorithms detailed in Chapter 2, the SEQUITUR algorithm has demonstrated the ability to discover hierarchical structure in sequence data, and has been altered to perform unsupervised segmentation tasks [Nevill-Manning and Witten (1997); Cohen et al. (2007)]. The main use of SEQUITUR, however, is not to segment sequences, but to compress them. It does so by iteratively extracting repeated subsequences in a time series, and replacing them with a lexical symbol. When the process is complete, the result is a significantly compressed sequence of symbols, and a grammar for expanding the sequence back into its original form. It's possible to treat each high-level symbol as representing a "chunk," and thereby treat this compression as a segmentation of the original sequence. However, the segmentation performance is significantly inferior to that of *VE* [Cohen et al. (2007)].

5.3 Hierarchical Voting Experts

The original application of *VE* was to segment text that had been stripped of punctuation and spaces. The implementation took sequences of characters as input and chunked them together to produce strings. However, the model for segmentation is a general one, and the implementation can be extended to work in more general domains. The most natural extension is to segment any sequence of tokens instead of just characters. In order to efficiently build and use an n-gram trie the tokens must be comparable, and it must be possible to impose a total ordering on them. Assuming this is the case, the same information theoretic metrics can be used by each expert to induce boundaries between tokens in the sequence.

The resulting chunks, then, are not strings but sequences of tokens. Notice that it is possible to compare sequences of tokens lexicographically in the same way we compare strings lexicographically based on their characters.

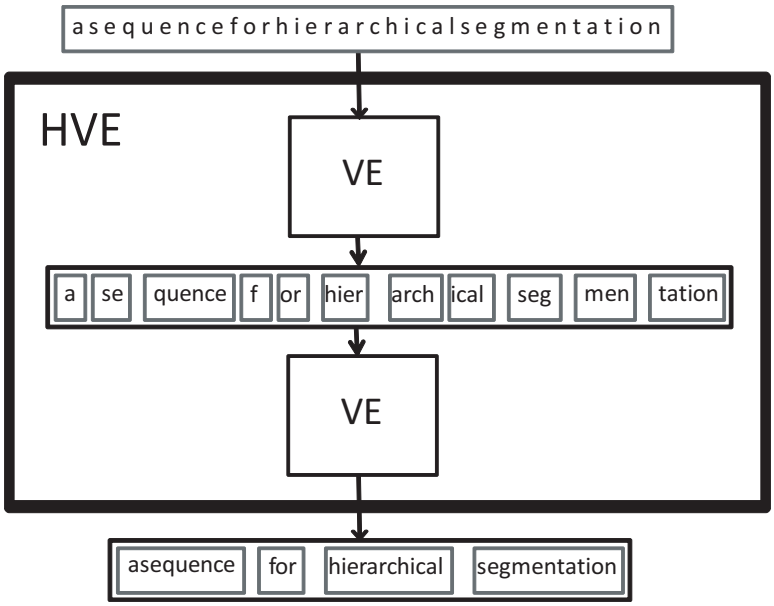

Figure 5.1 An example of 2-level *HVE* segmentation. *VE* is applied twice, the second time treating the chunks of the first iteration as tokens. This can be repeated arbitrarily many times.

The extension to Hierarchical Voting Experts (*HVE*) is then natural. Generalized *VE* is run on a sequence of tokens to obtain a sequence of chunks, each chunk composed of a short sequence of tokens. Those chunks are treated as the tokens of a new sequence, which can be chunked to create chunks of chunks. To do this, generalized *VE* is run again on the new sequence, building the n-gram trie by imposing a lexicographical ordering on the chunks. The experts use the trie to vote on how to split the sequence of chunks, and boundaries are induced in the same way as on a set of tokens. This process can be repeated indefinitely for any number of hierarchical layers (see Figure 5.1).

Consider text as an example. Suppose we created a mapping from each letter to a random three digit integer, and then translated a piece of text by replacing each letter with its three digits (see Figure 5.2). This sequence would have a simple hierarchical structure. Three digit subsequences could be grouped into letters, and the letters could be grouped into words. If we ran two level *HVE* on this data it would first segment the digit sequence in just the same way

as *VE*. Assuming it was successful, it would produce a sequence of chunks, each one of them containing three integer tokens. *HVE* would then run *VE* again on this sequence of chunks. The second level of *HVE* would produce a sequence of chunks of chunks of tokens. Ideally they would correspond to the words of the original text. In fact, this experiment was performed, and the results are included as Experiment 4.

HVE is a general extension of *VE* that can accept any sequence of comparable tokens as input. It is not at all necessary to use characters as the fundamental tokens. Any type of object that can be ordered and compared is valid. This includes RGB pixels or class labels or intensity values from a Fourier transform of an audio signal. It is one step further toward a general chunking algorithm. However, it is limited to one-dimensional ordered sequences with low token noise. So it is not a truly general solution, but it is a step in the right direction.

5.4 Experiments with HVE

Several experiments were designed to test the *HVE* algorithm. They demonstrate that the application of *HVE* to hierarchical data can induce accurate segmentation at each level.

However, the experiments also show that *HVE* is sensitive to the structure of its input. *HVE* segments a sequence based on the assumption that the true subsequences will be marked with low internal information, and their boundaries will be marked with high entropy. Sequence data that does not follow this pattern will be inscrutable to any incarnation of *HVE*. Conversely,

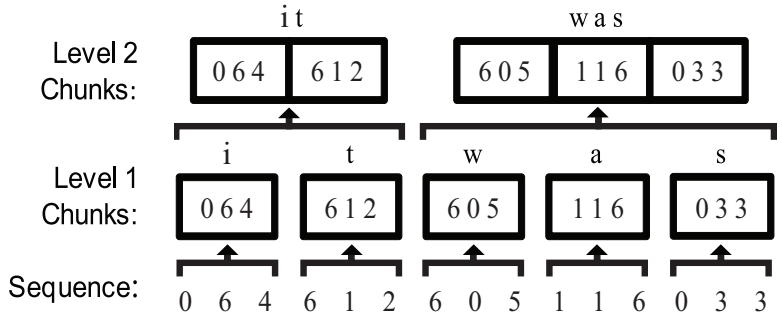

Figure 5.2 An illustration of hierarchical chunking. The digits are grouped into chunks that represent letters. Those chunks are then grouped into chunks that represent words.

the more the data conforms to this pattern the more successful *HVE* will be in segmenting it. This gives a clear theoretical delineation of the proper domain of this algorithm.

5.4.1 Dataset

The first 35,000 words (converted to lower case and stripped of punctuation and spaces) of George Orwell's novel "1984" were used as the base text for all of the experiments presented in this chapter. This was one of the benchmark datasets used to evaluate the original *VE* algorithm, so it was chosen for comparison [Cohen et al. (2007)]. To perform our experiments, this data was translated in several ways.

VE and *HVE* run in linear time with respect to the size of the dataset, and so they can be used to segment very long sequences [Cohen et al. (2007)]. However, the accuracy of the segmentation asymptotically approaches an upper bound, and there is little utility in using a corpus much larger than the base dataset. Additionally, the translations multiplied the length of the sequence, causing it to approach a million characters for some experiments. Even so, each experiment took only 5 to 10 minutes to run on a standard desktop computer.

5.4.2 Metrics

Three different metrics were used to evaluate the segmentation results for each experiment: f-measure, accuracy, and hit rate. These are the same metrics used to evaluate the original *VE* algorithm [Cohen et al. (2007)]. They are defined as follows. Let n be the number of correctly induced boundaries and let m be the total number of induced boundaries and let c be the total number of true boundaries in the sequence, then the accuracy of the segmentation is given by $a = n/m$ and the hit-rate is given by $h = n/c$. The f-measure of the segmentation is then defined to be $f = 2ah/(a + h)$. The value of the f-measure ranges from 0 to 1, 1 being perfect segmentation. It was chosen because it strikes a balance between measuring the accuracy of the induced boundaries, and the overall percentage of true boundaries that are found. The f-measure on the original text data set for single level *HVE* is 0.776. This value will be used as the baseline for evaluation of all second level segmentation, and it is included in each of the

pertinent tables. If the first level segmentation is perfect, we would expect the second level segmentation to have the same f-measure as the base text.

The choice of threshold V_t in step three of the *VE* algorithm has an effect on these metrics. Specifically, as V_t is raised, it takes more votes to induce a split. This causes the accuracy of the segmentation to go up, and the hit rate to go down. Conversely, lowering V_t generally lowers the accuracy and raises the hit rate. In each of our experiments we chose the V_t that struck a balance between the two. This method was also used in the original *VE* algorithm [Cohen et al. (2007)]. We hope to eventually find a principled way to set V_t, so that the algorithm needs no hand-tuning.

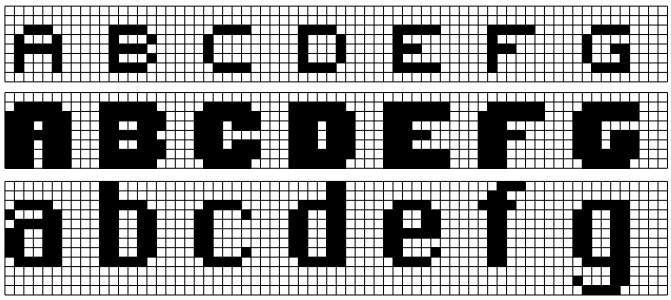

Figure 5.3 Fonts 1, 2 and 3 used in Experiment 1. Fonts 1 and 2 both have resolution 8x8 for each letter, and font 3 has resolution 12x8. However, even though fonts 1 and 2 have the same resolution, font 2 is more complex than font 1, in that each letter is composed of more unique pixel columns.

Experiment 1

We simulated the process of optically scanning the text to see if *HVE* could segment out the letters and then group them into words. For each character we considered each vertical column of black and white pixels in order, from left to right, ignoring all white space. A pixel column can be represented as a sequence of bits, which can be translated into a unique integer. We replaced each character in the text with the sequence of integers corresponding to the sequence of vertical columns of pixels that compose that character. Each integer became

a fundamental token of the sequence, as if the pixel column was viewed as a single token by the algorithm (see Figure 5.4). Three fonts were used, whose resolution and complexity varied (see Figure 5.3). For each font the text was transcribed and then two-level *HVE* was run on the translated data to segment the letters and then the words. The results demonstrate that *HVE* successfully segmented the sequence at both levels of the hierarchy (see Table 5.1).

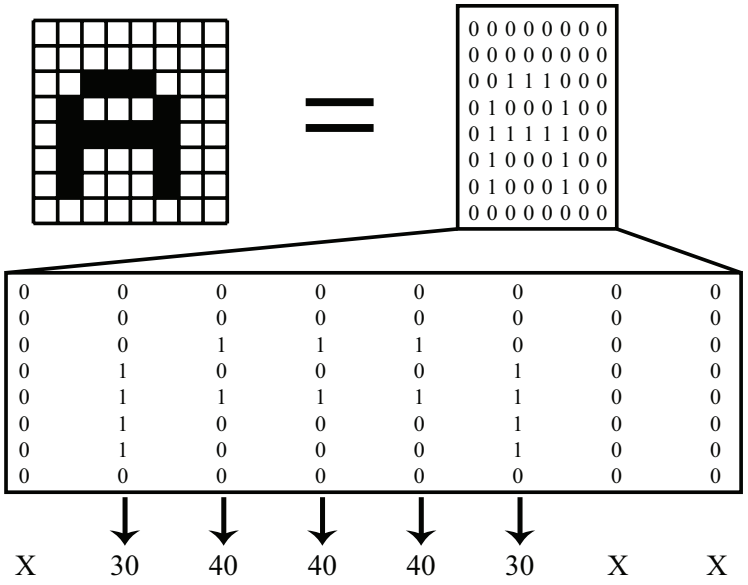

Figure 5.4 An illustration of the conversion of the letter "a" to a sequence of integers corresponding its pixel columns. The pixels are converted to bits, and each column of bits is converted to a decimal number. The white space is removed before and after each letter so that there are no boundary markers in the sequence.

5.4.3 Strengths and Limitations of HVE

A secondary goal of this chapter is to find the limitations of *HVE*, in order to clearly delineate its proper domain of applicability. The following experiments were designed to illustrate why and how *HVE* can fail to yield an accurate segmentation. Understanding its performance in these cases will allow us to understand what kinds of problems it should not be used to

Table 5.1 Experiment 1: Pixels to Characters to Words. The results are shown for each font at both levels of segmentation. The baseline segmentation is included for comparison.

	Font	F-measure	Accuracy	Hit Rate
Level 2	Font 1: 8x8	0.551	0.533	0.569
	Font 2: 8x8	0.742	0.734	0.750
	Font 3: 12x8	0.754	0.750	0.758
	Baseline	0.776	0.756	0.797
Level 1	Font 1: 8x8	0.751	0.739	0.763
	Font 2: 8x8	0.931	0.943	0.918
	Font 3: 12x8	0.972	0.985	0.959

solve. Table 5.2 illustrates the translation applied to the dataset for each of these experiments. The results are shown in Table 5.3, and are analyzed in Section 5.5.

Table 5.2 Translations from the original text to hierarchical sequences for experiments 2-4.

Letter	i	t	w	a	s
Morse Code	0 0	1	0 1 1	0 1	0 0 0
ASCII Octal	1 5 1	1 6 4	1 6 7	1 4 1	1 6 3
Random Octal	0 6 4	6 1 2	6 0 5	1 1 6	0 3 3

Experiment 2

Each character in the text was translated into its Morse code representation. The Morse code was represented with a sequence of bits - 0 corresponded to "dit" and 1 corresponded to "dah." Two-level *HVE* was run on the data to attempt to segment the Morse code into letters, and then the letters into words.

Experiment 3

Each character in the text was translated into its standard ASCII octal representation. So each character was translated into 3 digits ranging from 0 to 7. Two-level *HVE* was run on the translated data.

Experiment 4

A mapping from each letter to a random three digit octal number was generated. This mapping was used to translate the base text by replacing each character in it with its corresponding three digits. Two-level *HVE* was run on this data to segment the letters and then the words.

Table 5.3 Experiment 2-4: The results for each experiment are shown for the first and second level of segmentation. The baseline is included for comparison.

	Dataset	F-measure	Accuracy	Hit Rate
Level 2	Morse Code	0.100	0.107	0.095
	ASCII Octal	0.028	0.030	0.027
	Random Octal	0.743	0.768	0.719
	Baseline	0.776	0.756	0.797
Level 1	Morse Code	0.397	0.444	0.358
	ASCII Octal	0.254	0.261	0.247
	Random Octal	0.944	0.978	0.913

5.4.4 Phoneme Segmentation

As mentioned in Chapter 2, most segmentation algorithms that have been applied to natural language have been used to segment phonemic translations of audio speech. It is, in some sense, unfair to compare the results of *HVE*'s segmentation of text with other algorithms segmentation of phoneme or syllable sequences. Text is not a naturally occurring sequence. It is an encoding of a naturally occurring sequence - spoken language. The encoding might obscure the information theoretic signatures in the same way Morse code or ASCII octal might. At the very least we shouldn't expect any care to be taken to preserve them. Therefore, it should be at least as easy to segment sequences of phonemes into words as it is to segment sequences of text.

The spellings of words are cobbled together over the centuries through the converging and diverging of many different languages, with words being borrowed from here and there as necessity dictates. Conversely, if humans use these information theoretic signatures to segment

speech, we would expect the sounds of speech to evolve concordantly. The sounds of words are more than an encoding of ideas, they are a natural and organic part of the way we think. We would expect that our language should evolve to make the segmentation task easier. The phonetic sequences should demonstrate the markers required for segmentation more clearly than text, which has not had the time or the pressure to adopt these markings. Presumably, it is disadvantageous if children have a hard time learning to segment their native language, but not nearly so bad if the same is true of their written words. After all, words are written with spaces between them, so the child need never learn to segment them, should they be written without. Therefore it is interesting to compare the performance of the algorithm in both domains.

Experiment 5

The CMU Pronouncing Dictionary was used to translate each word in the dataset into its phonemic representation. The CMU Dictionary uses a text base representation of 39 phonemes to represent over 125,000 words. Each word in the dataset was replaced with its corresponding phonetic representation. For instance, the opening phrase of Orwell's 1984 "It was a bright cold day in April," became "ih1 t * w aa1 z * ah0 * b r ay1 t * k ow1 l d * d ey1 * ah0 n * ey1 p r ah0 l" (delimiters added for clarity). A few words in the dataset were not present in the dictionary, including some of the proper names and some words invented by George Orwell. These were omitted from the translated text. One-level *HVE* was run on the translated data. Table 5.4 summarizes the results which are analyzed in the next section.

Table 5.4 Phoneme Results: Note that the segmentation of phonemes to words is slightly better than the baseline segmentation of text to words.

Dataset	F-measure	Accuracy	Hit Rate
Phonemes	0.807	0.808	0.806
Baseline	0.776	0.756	0.797

5.5 Analysis of Results and Discussion of *HVE*

It is clear that *HVE* is able to perform higher order segmentation of hierarchically structured data, given that the data exhibits the necessary information theoretic markers. In Experiment 1 (font conversion) the first order segmentation f-measure increased as the complexity of the font increased. In particular, it performed very well on fonts 2 and 3. As expected, the second order segmentation approached the baseline in both of these cases. The first order segmentation for font 1 was not as good, and as a result the second order segmentation was worse. However, even though the level 1 segmentation was only 74% accurate, the f-measure for level 2 was still .551. This indicates that the algorithm it at least partially robust to segmentation noise between levels. And we can see with font 2 that, once the level 1 segmentation accuracy reaches roughly 90%, the level 2 segmentation is very close to the baseline. It seems that the algorithm can compensate for a small amount of segmentation error in the lower level.

In Experiments 2 and 3 (Morse code and ASCII), *HVE* had considerably poor performance and was unable to find the letter boundaries at the lowest level of segmentation. Naturally, the second level segmentation was also very poor. It is important to understand why this happened in order to understand the limitations of this model.

In Experiment 2 there was too much ambiguity in the sequence. Every binary string of length less than 5 has a meaning in Morse code. Given a sequence of dits and dahs, it is simply indeterminate where the breaks should go. In practice, Morse code is not sent in one continuous stream. The sender places short pauses between each letter, and longer pauses between each word. Thus the receiver does not have to perform a segmentation task, but only a translation. The algorithm was not given these breaks, but was asked to induce them. However, as stated, virtually any segmentation of the Morse code would have induced legal letters. In *HVE*'s terms, this means that the internal entropy of the true subsequences is no lower than the internal entropy of false subsequences of roughly the same length. *HVE* relies on the assumption that most combinations of symbols of a given length are not a proper chunk, and are in fact random noise. Non-chunks should have high internal entropy - they should be uncommon. The true subsequences should occupy only a small subset of the total space of

possible subsequences. But in this case all sequences of the appropriate length could have been a chunk. The "low internal information" marker was not present, so the algorithm was unable to segment the sequence.

In Experiment 3 the ASCII octal subsequences did, in fact, have low internal entropy compared with the false subsequences. Only 26 of the possible 512 three digit octal numbers are used to map lowercase letters. So most three digit combinations had very high entropy (appeared infrequently) compared to the ones associated with characters. However, the octal representation of the ASCII character set for English lowercase letters ranges from 141 to 172. Every number starts with a 1. This means that at the end of each three digit number in the translated text, it is always certain which digit is coming next. It will be the leading "1" from the next letter's representation. This means that the boundary entropy at the end of each octal number is 0. The "high boundary entropy" marker was not present, so the algorithm was unable to segment the sequence.

In Experiment 4 an octal representation was still used for each character, except the mapping was generated randomly instead of being taken from the ASCII table. The internal information of the correct subsequences was just as low as in Experiment 3, but the boundary entropy was much higher. Randomly distributing the character representations through the domain disambiguated them. Accordingly, *HVE* was able to segment the lower level with a high accuracy and hit rate. And since that segmentation was accurate, the second order segmentation's performance approached the baseline segmentation of the original text (see Table 5.3).

In Experiment 5 the segmentation of the phonemes is slightly better than the baseline segmentation of the original text. This lends a little bit of credence to the idea that the signatures used by *VE* are more salient in spoken speech than in printed text. Moreover, the results reported here are much better than those in Chapters 3 and 4, illustrating the inherent difficulty of working with audio. However, if an acoustic model could be trained to transcribe speech into phonemes as accurately as the CMU phonetic dictionary, we might expect similar results.

5.6 HVE-3E

In addition to extending *VE* to work on hierarchical data, a mechanism was devised to use higher order model information to increase segmentation accuracy at the lower level. To do this standard *HVE* was run on a given sequence to obtain a sequence of chunks. Then the second order model (ngram trie) was built using those chunks. Finally, the algorithm was re-run on the original sequence with the addition of a third voting expert. This modification of the algorithm is called *HVE-3E*, where 3E stands for 3 voting Experts.

The third voting expert uses the higher order model to help split the lower order sequence. For each position of the sliding window, it checks whether any subsequence starting at the beginning of the window matches one or more chunks known to the higher order model. If so, it votes to place a break after the most common of those subsequences (see Figure 5.5). If no match is found, it does not vote. After the third expert has added its votes to those of the first and second experts, the sequence is split based on the cumulative votes. When inducing the split, it is necessary to raise the threshhold V_t by one to accomodate the additional votes. This process can be repeated at each level of the hierarchy. So, when using the third voting expert, segmentation must be done twice at each level - once to build a temporary second order model for use by the third expert, and a second time to produce the final segmentation.

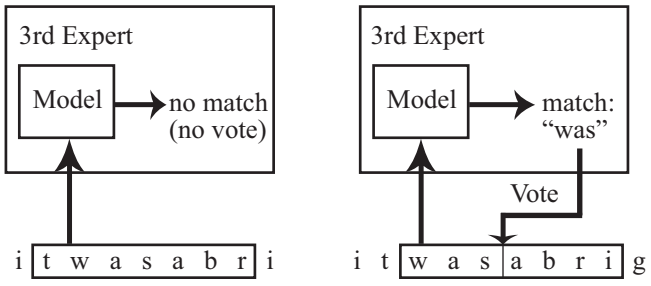

Figure 5.5 An illustration of the third voting expert. Given a sliding window, it tries to find a sequence that its model recognizes, starting at the beginning of the window. If it doesn't, it does not vote. If it does, it votes to place a break after that sequence. This vote is combined with those from the other two experts.

The rationale behind the third voting expert is that, after building a higher order model, the most common tokens in that model will correspond to true common segments in the lower level sequence. So the third expert can recognize sequences that commonly become chunks, and vote to reinforce them. This reinforcement improves the overall segmentation accuracy.

Experiment 6

To test the third voting expert Experiments 1 and 4 were re-run with the additional expert added. The third expert was used to increase segmentation accuracy on both the first and second level of the hierarchy. Table 5.5 demonstrates the improvement at both levels.

Table 5.5 Experiment 6: *HVE-3E*. Compare with Table 5.1 and Table 5.3. The % Change of the F-measure is included for comparison.

	Dataset	F-measure		Accuracy	Hit Rate
		Result	% Change		
Level 2	Font 1: 8x8	0.642	+16.5%	0.614	0.672
	Font 2: 8x8	0.772	+4.0%	0.771	0.773
	Font 3: 12x8	0.768	+1.9%	0.756	0.781
	Random Octal	0.775	+4.3%	0.776	0.781
Level 1	Font 1: 8x8	0.806	+7.3%	0.795	0.817
	Font 2: 8x8	0.959	+3.0%	0.999	0.921
	Font 3: 12x8	0.974	+0.2%	0.989	0.959
	Random Octal	0.972	+3.0%	0.992	0.959

5.6.1 Analysis of Results and Discussion of *HVE-3E*

Experiment 6 clearly shows the improvements gained from the addition of the third voting expert. They are small, but present at each level of segmentation, and for each data set. This shows that it is possible to improve lower level segmentation by using information from higher order models. Additionally, the improvement is more substantial on the more difficult data sets. It seems that there is more to gain from reinforcing common chunks on difficult sequences.

More generally we would expect the higher order model to be able to help bootstrap the lower level in many ways. *HVE-3E* is a very simple implementation of this idea, in that it only finds exact matches inside the current window. It is expected that more sophisticated methods of propagating information back down the hierarchy could improve segmentation even more.

Cheng and Mitzenmacher have described a more sophisticated third expert which improved segmentation accuracy slightly more than this one [Cheng and Mitzenmacher (2005)]. Their algorithm, however, is not hierarchical, and is much more complex than our third expert. Additionally, both experts are compatible, *i.e.*, they could be used simultaneously to increase accuracy even further. In any case, we already see improvements at each level of the hierarchy with simple chunk matching, demonstrating that information from higher order models can be put to good use increasing the segmentation accuracy at lower levels.

5.7 Summary

This chapter described a natural extension of the Voting Experts (*VE*) algorithm [Cohen et al. (2007)], called Hierarchical Voting Experts (*HVE*), which segments hierarchically structured sequences. It was shown that *HVE* can successfully perform hierarchical segmentation on a variety of datasets. Also, it was demonstrated that *HVE* is sensitive to information theoretic features of the dataset. Specifically it requires that the information theoretic signatures of chunks be present and unobscured. A technique for improving the segmentation accuracy by making use of higher order models was also demonstrated, and it was effective at each level of the hierarchy.

It seems that many of the domains in which we would like to be able to apply this kind of algorithm naturally exhibit the signatures of chunks. Therefore, the possible applications of the *HVE* chunking model are numerous. A general method for the segmentation of any kind of sensory data based on internal and boundary entropy would presumably be a powerful and fascinating model. We are nowhere close to such an algorithm, but we are interested in its feasibility and possible applications. In any case, the *VE* model has continually proven itself powerful, and it deserves further study.

CHAPTER 6. SUMMARY AND DISCUSSION

In this thesis I have described a technique for the unsupervised segmentation of acoustic speech using the Voting Experts algorithm. To my knowledge, this is the first unsupervised algorithm of this sort. In particular, the previous work in this field has been focused on the segmentation of text or textual phonetic transcripts of speech. In Chapter 3 I showed that the *VE* model is capable of inducing an accurate segmentation on an audio stimulus stream with very limited training data. Specifically, the results of a famous series of infant speech segmentation experiments were reproduced.

Additionally, the same algorithm was used to segment a very large natural language speech stream (an audio book). The resulting segmentation was evaluated by human graders, and was determined to be significantly more accurate than chance. This demonstrated that the algorithm is capable of inducing breaks in real spoken language, and not just artificially generated speech. In other words, the information theoretic markers used by *VE* really are present in natural language streams, and can be used to find word breaks. However, the evaluation of that particular dataset only showed that the algorithm worked better than chance. It gave no objective measure of the segmentation quality, or how useful that segmentation might be for the purpose of language learning. This was an unfortunate consequence of using a large, unlabeled natural language dataset.

In Chapter 4 an improved acoustic model was suggested, which was more closely related to modern speech recognition systems. It was implemented and shown to significantly improve the performance of *VE*'s segmentation on the infant datasets. The original experiments were repeated using the new model, and a much more thorough evaluation of the algorithm's performance and its sensitivity to its parameters was performed. This evaluation conclusively

demonstrated *VE*'s ability to segment acoustic speech streams, as long as they contain the requisite statistical cues.

In Chapter 5 the Voting Experts algorithm itself was generalized and extended to Hierarchical Voting Experts. This extension proved able to segment hierarchically structured sequences, such as vertical pixel columns extracted from text. It was also thoroughly tested to demonstrate its limitations, and to identify situations where it fails to produce a reliable segmentation. Finally, a third voting expert was introduced that used higher order models to improve the segmentation at lower levels of *HVE*. This new algorithm *HVE-3E* showed significant improvement in the segmentation quality at each level of the hierarchy.

So now we can see the progression of the work. A basic application of Voting Experts to acoustic streams induces segmentations with accuracy significantly greater than chance, both on artificially generated and spoken language. This segmentation can be improved by improving the acoustic model. *VE* itself can be extended and improved as well.

This specifies two independent directions for future work on this topic. The first is developing better unsupervised acoustic models. Given the maturity of speech recognition research, this is most certainly possible. The second is extending *VE* to improve its performance. The *VE* model is not married to the sliding window implementation that uses the experts. Its model is simply the intuition that words exhibit low internal information, and high entropy at their boundaries. It seems likely that this intuition could admit of many incarnations, some of which might be better suited to acoustic segmentation. Moreover, several strategies for the improvement of segmentation were discussed and then dismissed in Chapter 2, since they were more appropriate for bootstrapping than beginning the process. However, there is no reason that *VE* couldn't be augmented with this same bootstrapping.

The results of this work are encouraging, and lay the foundation for future investigation. For now, we know that the acoustic models and algorithms presented here are capable of beginning the language segmentation process. They do not constitute a complete model of the infant segmentation mechanism, but that was never the goal. The goal was a practical algorithm inspired by infant speech segmentation. The algorithms presented here are clearly

on that path, even if they haven't reached full maturity.

It's unknown how accurately a purely distributional algorithm could segment natural speech streams. Perhaps the best it could do is to produce breaks with barely enough accuracy to learn phonotatic rules, or to uncover some other, more reliable cue. Perhaps it could induce a near perfect segmentation, rendering the other cues meaningless. Neither of these possibilities seems very likely. The only way to find out for sure is to develop and refine distributional segmentation algorithms, and see how well they can perform. The algorithms presented in this thesis represent an initial baseline. Their performance is surprisingly good given their simplicity and the inherent difficulties associated with acoustic speech streams. They are a first step, small but definite, toward the unsupervised acquisition of language. Moreover, they are a first step with the promise of many more. If nothing else, this work demonstrates the potential of the statistical segmentation of audio. As with any proper scientific development, it is incremental, certain and thorough. But it also exhibits the most desirable trait that any new advancement might admit - it is inspirational. It calls attention to an area of dramatic potential. Hopefully this work will encourage further research along these lines, and yield other useful, powerful and innovative results.

BIBLIOGRAPHY

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

Ando, R. K. and Lee, L. (2000). Mostly-unsupervised statistical segmentation of Japanese: applications to Kanji. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 241–248.

Aslin, R. N. (1996). *Models of word segmentation in fluent maternal speech to infants*. Signal to Syntax: Bootstrapping from Speech to Grammar in Early Acquisition. Erlbaum.

Blanchard, D. and Heinz, J. (2008). Improving word segmentation by simultaneously learning phonotactics. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, page 6572, Manchester, England.

Bortfeld, H., Morgan, J. L., Golinkoff, R. M., and Rathbun, K. (2005). Mommy and me: Familiar names help launch babies into speech-stream segmentation. *Psychological Science*, 16(4):298–304.

Brandl, H., Wrede, B., Joublina, F., and Goerick, C. (2008). A self-referential childlike model to acquire phones, syllables and words from acoustic speech. In *Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL)*, pages 31–36.

Brent, M. R. (1999). An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34(1-3):71–105.

Brent, M. R. and Cartwright, T. A. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

Brent, M. R. and Siskind, J. M. (2001). The role of exposure to isolated words in early vocabulary development. *Cognition*, 81:B33–B44.

Cairns, P. and Shillcock, R. (1997). Bootstrapping word boundaries: A bottom-up corpus-based approach to speech segmentation. *Cognitive Psychology*, 33:111–153.

Cheng, J. and Mitzenmacher, M. (2005). The markov expert for finding episodes in time series. In *DCC '05: Proceedings of the Data Compression Conference*, pages 454–454.

Chomsky, N. (1955). The logical structure of linguistic theory. MIT Humanities Library. Microfilm. Published in 1977 by Plenum.

Christiansen, M., Allen, J., and Seidenberg, M. (1998). Learning to segment speech using multiple cues: A connectionist model. *Language and Cognitive Processes*, 12(2-3):221–268.

Church, K. W. (1987). *Phonological parsing in speech recognition*. Kluwer Academic Publishers, Norwell, MA, USA.

Cohen, P., Adams, N., and Heeringa, B. (2007). Voting Experts: An unsupervised algorithm for segmenting sequences. *Journal of Intelligent Data Analysis*, 11(6):607–625.

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(4):357–366.

de Marcken, C. (1995). The unsupervised acquisition of a lexicon from continuous speech. Technical Report AIM-1558.

Dittenbach, M., Merkl, D., and Rauber, A. (2000). The growing hierarchical self-organizing map. *Proceedings of the IEEE-INNS-ENNS IJCNN 2000*, 6:15–19.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Fiser, J. and Aslin, R. (2005). Encoding multielement scenes: statistical learning of visual feature hierarchies. *J Exp Psychol Gen*, 134(4):521–537.

Fiser, J. and Aslin, R. N. (2002). Statistical learning of higher-order temporal structure from visual shape sequences. *Journal of Experimental Psychology*, 28(3):458–467.

Fleck, M. M. (2008). Lexicalized phonotactic word segmentation. In *46th Annual Meeting of the ACL*, pages 130–138, Morristown, NJ. ACL.

Fritzke, B. (1995). Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13.

Gambell, T. and Yang, C. (2008). Mechanisms and constraints in word segmentation. Manuscript, Yale University.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallet, D., and Dahlgren, N. (1990). The DARPA TIMIT acoustic-phonetic continuous speech corpus CDROM. NITS order number pb91-505065, October.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 673–680, Morristown, NJ.

Goodsitt, J. V., Morgan, J. L., and Khul, P. K. (1993). Perceptual strategies in prelingual speech segmentation. *Journal of Child Psychology*, 20:229–252.

Halle, M. (1978). Formal vs. functional considerations in phonology. *Studies in the Linguistic Sciences Urbana, Ill.*, 8(2):123–134.

Harris, Z. S. (1955). From phoneme to morpheme. *Language*, 31:190–222.

Hayes, J. R. and Clark, H. H. (1970). *Experiments in the segmentation of an artificial speech analogue*. Cognition and the Development of Language. Wiley, New York.

Iwahashi, N. (2006). *Symbol Grounding and Beyond*, volume 4211/2006 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg.

Jusczyk, P. W. (1999). How infants begin to extract words from speech. *Trends in Cognitive Sciences*, 3(9):323 – 328.

Jusczyk, P. W., Culter, A., and Redanz, N. J. (1993a). Infants' preference for the predominant stress patters of english words. *Child Development*, 64(3):675–687.

Jusczyk, P. W., Friederici, A. D., Wessels, J., Svenkerud, V. Y., and Jusczyk, A. M. (1993b). Infants' sensitivity to the sound patterns of native language words. *Journal of Memory and Language*, 32:402–420.

Jusczyk, P. W., Hohne, E. A., and Bauman, A. (1999). Infants' sensitivity to allophonic cues for word segmentation. *Perception & Psychophysics*, 61(8):1465–1476.

Jusczyk, P. W., Luce, P. A., and Charles-Luce, J. (1994). Infants' sensitivity of phonotactic patterns in the native language. *Journal of Memory and Language*, 33:630–645.

Kirkham, N. Z., Slemmer, J. A., and Johnson, S. P. (2002). Visual statistical learning in infancy: evidence for a domain general learning mechanism. *Cognition*, 83:B35–B42.

Klatt, D. H. (1979). Speech perception: A model of acoustic-phonetic analysis and lexical access. *Journal of Phonetics*, 7:279–285.

Kohonen, T. (1988). Self-organized formation of topologically correct feature maps. *Neurocomputing: foundations of research*, pages 509–521.

Krakow, R. A. (1999). Physiological organization of syllables: a review. *Journal of Phonetics*, 27:23–54.

Liberman, A. M., Cooper, F. S., Shankweiler, D. P., and Studdert-Kennedy, M. (1967). Perception of the speech code 1. *Psychological Review*, 74(6):431–461.

Mattys, S. L. and Jusczyk, P. W. (2001). Do infants segment words or recurring contiguous patterns? *Journal of Experimental Psychology*, 27(3):644–655.

Mattys, S. L., White, L., and Melhorn, J. F. (2005). Integration of multiple speech segmentation cues: a hierarchical framework. *Journal of Experimental Psychology*, 134(4):477–500.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Miller, M. and Stoytchev, A. (2008a). Hierarchical Voting Experts: An unsupervised algorithm for hierarchical sequence segmentation. In *Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL)*.

Miller, M. and Stoytchev, A. (2008b). Unsupervised audio speech segmentation using the Voting Experts algorithm. In *NIPS 2008 workshop on Speech and Language: Learning-based Methods and Systems*, Whistler, British Columbia, Canada.

Miller, M. and Stoytchev, A. (2008c). Unsupervised audio speech segmentation using the Voting Experts algorithm. In *NIPS Workshop on Speech and Language: Learning-based Methods and Systems*.

Miller, M., Wong, P., and Stoytchev, A. (2009). Unsupervised segmentation of audio speech using the Voting Experts algorithm. In *Proceedings of the Second Conference on Artificial General Intelligence (AGI)*, Arlington, Virginia.

Nevill-Manning, C. and Witten, I. (1997). Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, pages 7:67–82.

Peters, A. M. (1983). *The units of language acquisition*. Cambridge Univ. Pr., Cambridge, New York.

Pinker, S. (1984). *Language Learnability and Language Development*. Harvard University Press, Cambridge, Mass.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. *Readings in speech recognition*, pages 267–296.

Roy, D. and Pentland, A. (2002). Learning words from sights and sounds: a computational model. *Cognitive Science*, 26:113–146.

Saffran, J. R., Aslin, R. N., and Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928.

Saffran, J. R., Johnson, E. K., Aslin, R. N., and Newport, E. L. (1999). Statistical learning of tone sequences by human infants and adults. *Cognition*, 70:27–52.

Saffran, J. R., Newport, E. L., Aslin, R. N., and Tunick, R. A. (1997). Incidental language learning: Listening (and learning) out of the corner of your ear. *Psychological Science*, 8(2):101–105.

Shannon, C. (1951). Prediction and the entropy of printed english. Technical report, Bell System Technical Journal.

Sinapov, J., Wiemer, M., and Stoytchev, A. (2009). Interactive learning of the acoustic properties of household objects. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan.

Swingley, D. (2005). Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*.

Venkataraman, A. (2001). A statistical model for word discovery in transcribed speech. *Computational Linguistics*, 27(3):352–372.

Walker, W., Lamere, P., Kwok, P., Raj, B., Gingh, R., and Gouvea, E. (2004). Sphinx-1: A flexible open source framework for speech recognition. Technical Report TR-2004-139.

Wolff, J. G. (1977). The discovery of segments in natural language. *British Journal of Psychology*, 68:97–106.