Unsupervised Audio Speech Segmentation Using the Voting Experts Algorithm

Matthew Miller
Developmental Robotics Laboratory
Iowa State University
mamille@iastate.edu

Alexander Stoytchev
Developmental Robotics Laboratory
Iowa State University
alexs@iastate.edu

Abstract

Human beings have an apparently innate ability to segment continuous audio speech into words, and that ability is present in infants as young as 8 months old. This propensity towards audio segmentation seems to lay the groundwork for language learning in human beings. To artificially reproduce this ability would be both practically useful and theoretically enlightening. In this paper we propose an algorithm for the unsupervised segmentation of audio speech, based on the Voting Experts (*VE*) algorithm, which was originally designed to segment sequences of discrete tokens into categorical episodes. We demonstrate that our procedure is capable of inducing breaks with an accuracy substantially greater than chance, and suggest possible avenues of exploration to further increase the segmentation quality. We also show that this algorithm can reproduce results obtained from segmentation experiments performed with 8-month-old infants.

1 Introduction

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 [1]. Presumably, the language learning process begins with learning which utterances are tokens of the language and which are not. Several experiments have shown that this segmentation is performed in an unsupervised manner, without requiring any external cues about where the breaks should go [1][2]. Saffran and others have suggested that humans use statistical properties of the audio stream to induce segmentation. An accurate characterization of this ability would presumably be a theoretical and practical breakthrough in automatic language processing. Along those lines, this paper proposes a method for the unsupervised segmentation of spoken speech, based on an algorithm designed to segment discrete time series into meaningful episodes. We suggest that our model may capture the human process of segmentation in some small way. To substantiate our claim, we replicate an experiment that was performed on 8 month old infants, and show that our algorithm performs similarly to the children.

Paul Cohen has suggested an unsupervised algorithm called Voting Experts (*VE*) that uses the information theoretical properties of *internal entropy* and *boundary entropy* to segment discrete time series into categorical episodes [3]. *VE* has previously demonstrated, among other things, the ability to accurately segment plain text that has had the spaces and punctuation removed [4]. More generally, Cohen has suggested that the information theoretic model of *VE* may be a useful model for human chunking, *i.e.*, human beings might be using these same information theoretical markers to segment sensory data. In this paper we extend *VE* to work on audio data. The extension is not a trivial or straightforward one, since *VE* was designed to work on sequences of *discrete* tokens and audio speech is continuous. We then use this algorithm to reproduce Saffran *et al.'s* original experiments [1] and show that *VE* can model their original results.

2 Related Work

Our work is directly inspired by the psychological studies of audio segmentation in human beings [1][2][5]. These studies show us that the unsupervised segmentation of natural language is possible, and does not require prohibitively long exposure to the audio stream. In the original infant experiments the infants were played a stimulus stream consisting of three-syllable nonsense words appearing in random order [1]. Those words were *tupiro*, *golabu*, *bidaku* and *padoti*. The stream was generated by a speech synthesizer and contained no audio cues as to the locations of the word boundaries. That is, there were no substantial pauses between words, or any variation in the tone or speed of the artificial voice. The only indication of the word breaks was the statistical relationships between the phonemes.

After the infants were acclimated to this audio stream, they were played a second stimulus stream consisting of a single three syllable word repeated over and over. In some cases that word was drawn from the original "language," and in other cases it was not. By observing the reaction of the infants, specifically their listening times, the experimenters were able to demonstrate that the infants had learned the words and word boundaries of the first stream.

This is an amazing conclusion, especially given the short duration of the stimulus streams and their lack of prosodic information. This gives us an insight into one way in which human beings begin to learn language. We use statistical properties of audio streams to break them into chunks. The goal of this paper is to model that process and to suggest that an artificial system might also be able to put that model to good use.

However, these studies do little to direct us towards a functioning algorithm capable of such a feat. Conversely, there are several related algorithms capable of segmenting categorical time series into episodes [6][7][8][9][10][11]. But these are typically supervised algorithms, or not specifically suited for segmentation. In fact, many of them have more to do with finding minimum description lengths of sequences than with finding logical segmentations.

As mentioned, this work makes use of the Voting Experts algorithm [3], which was designed to do with discrete token sequences what we are trying to do with real audio. That is, given a time series, specify all of the logical breaks to segment the series into categorical episodes. One major contribution of this paper is transforming an audio signal so that the VE model can be applied to it.

3 Overview of the Voting Experts Algorithm

The *VE* algorithm is based on the hypothesis that natural breaks in a sequence are usually accompanied by two information theoretic signatures [3][12]. These are low *internal entropy* of chunks, and high *boundary entropy* between chunks. A *chunk* can be thought of as a sequence of related tokens. For instance, if we are segmenting text, then the letters can be grouped into chunks that represent the words.

Internal entropy can be understood as the surprise associated with seeing a group of objects together. More specifically, it is the negative log of the probability of those objects being found together. Given a short sequence of tokens taken from a longer time series, the internal entropy of the short sequence is the negative log of the probability of finding that sequence in the longer time series. So the higher the probability of a chunk, the lower its internal entropy.

Boundary entropy is the uncertainty at the boundary of a chunk. Given a sequence of tokens, the boundary entropy is the expected information gain of being told the next token in the time series. This is calculated as $H_I(c) = -\sum_{h=1}^{m} P(h,c)log(P(h,c))$ where c is this given sequence of tokens, $P(h,c)$ is the conditional probability of symbol h following c and m is the number of tokens in the alphabet. Well formed chunks are groups of tokens that are found together in many different circumstances, so they are somewhat unrelated to the surrounding elements. This means that, given a subsequence, there is no particular token that is very likely to follow that subsequence.

In order to segment a discrete time series, *VE* preprocesses the time series to build an n-gram trie, which represents all its subsequences of length less than or equal to n. It then passes a sliding window of length n over the series. At each window location, two "experts" vote on how they would break the contents of the window. One expert votes to minimize the internal entropy of the induced chunks, and the other votes to maximize the entropy at the break. The experts use the trie to make these calculations. After all the votes have been cast, the sequence is broken at the "peaks"

- locations that received more votes than their neighbors, so long as the total votes at the location exceeded a threshold V_t. For all of our experiments we chose $n = 7$ and $V_t = 5$. This algorithm can be run in linear time with respect to the length of the sequence, and can therefore be used to segment very long sequences. For further technical details of *VE*, or a discussion of the roles of V_t and n, see the journal article [3].

It is important to emphasize the *VE model* over the actual implementation of *VE*. The goal of our work is to segment audio speech based on information theoretic markers, and to evaluate how well they work for this task. In order to do this, we use a particular implementation of Voting Experts, and transform the audio data into a format it can use. This is not necessarily the best way to apply this model to audio segmentation, but it is one way to approach the problem.

The model of segmenting based on low internal entropy and high boundary entropy is also closely related to the work in psychology mentioned earlier [2]. Specifically, they suggest that humans segment audio streams based on conditional probability. That is, given two phonemes A and B, we conclude that AB is part of a word if the conditional probability of B occurring after A is high. Similarly, we conclude that AB is not part of a word if the conditional probability of B given A is low. The information theoretic markers of *VE* are simply a more sophisticated characterization of exactly this idea. Internal entropy is directly related to the conditional probability inside of words. And boundary entropy is directly related to the conditional probability between words. This relationship motivates the claim that the *VE* model at least partially captures the human chunking process.

4 Datasets

We obtained two stimulus streams from the infant speech segmentation experiments performed by Saffran *et al.*[1]. Each audio stream is 60 seconds long and contains roughly 90 "words." The first stream (stream A) was composed, as described above, 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 similar to 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, we can perform a more sophisticated evaluation of our model since it produces explicit break locations. We found it more informative to test our 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 our 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. Many times the sound morphs from one allophone to the next, providing no clear boundary between them. However, the speech in the streams used by Saffran *et al.* is extremely regular, which allowed us to consistently place breaks at the same location in each word. The resulting "answer keys" were as accurate as possible, however the true breaks in a word are, after a certain point, a matter of opinion. This is a fundamental problem in the evaluation of any speech segmentation. We will discuss how we dealt with these problems in the section entitled Evaluation Methodology.

5 Audio Segmentation Algorithm

The following three steps are required in order to segment an audio stream using the *VE* algorithm. First, the audio stream must be temporally discretized. Second, those discretized values must be tokenized by labeling them based on some small alphabet. Finally, the *VE* algorithm can segment the token sequence. The induced breaks must then be translated back into temporal break locations in the original audio stream.

5.1 Discretization

We 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. This is a very standard procedure, and a technical explanation of each of these steps is available in the Sphinx documenta-

tion [13]. We performed the Fourier Transform 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 power spectrums, representing the intensity information in 257 frequency bins, taken every 10ms. These power spectrums can be viewed as a spectrogram representing the intensity information over time (see Figure 1).

5.2 Tokenization

The FFT converts the audio into a discrete sequence of continuous real-valued vectors. These vectors must then be tokenized. The method of tokenization will certainly affect the overall segmentation quality, and there are a broad range of techniques that could be used. We chose a fairly simple, unsupervised method that was found to work. This step could certainly be improved to the benefit of the overall segmentation. However, our aim is merely to test the model and demonstrate its potential. In order to do so, we trained a Self-Organizing Map (SOM) [14] on each audio stream's spectrogram information. Each time slice of the spectrogram was treated as a single instance. So the SOM was essentially used to cluster similar sounds together.

In order to avoid specifying the number of SOM nodes a priori, we used a Growing Grid SOM (GGSOM) for our experiments. A GGSOM is a self-organizing map that automatically grows to an appropriate size [15]. 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 our experiments we chose a $\tau = 0.05$. This lead to an SOM with 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. We used the implementation of a Growing Hierarchical SOM (GH-SOM) in the Java SOM Toolbox to train our Growing Grid [16].

After training a GGSOM on the spectrogram data we then used it to classify each time slice of the spectrogram. Each node in the SOM was given a unique label. Each time slice of the spectrogram was then labeled according to the node with the highest activation value for that time slice. So the clustering produced a sequence of node labels corresponding to each instance in the spectrogram (see Figure 1). In this way we produced a discrete sequence of tokens representing the audio data.

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 for similar sounds that were held for any length of time. In order to be time independent, we collapsed these repeated sequences into just one instance of the given node. This effectively denotes a prolonged period of the same sound by a single state (see Figure 1).

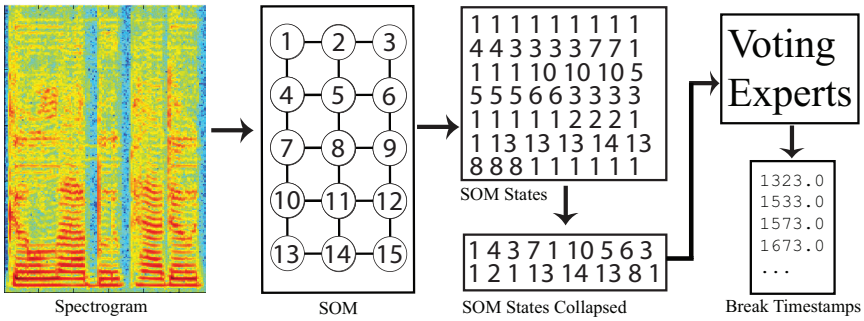

Figure 1: The audio segmentation process: The spectrogram of an audio stream is used to train an SOM, which is then used to label each time slice of the spectrogram. The repeated labels are removed, and that sequence is used to train the Voting Experts model, which then segments the sequence and specifies the timestamps of the induced breaks (in milliseconds).

5.3 Segmentation

In order to segment the tokenized sequences, we ran *VE* on the sequence of SOM states. *VE* placed breaks at locations of low internal entropy and high boundary entropy. Then, after accounting for the collapsed (*i.e.*, repeated) tokens, it produced the time stamps of all of the induced break locations in each audio stream. These breaks were then checked against the answer keys that had been manually created for each stimulus stream.

6 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. We counted an induced break 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 makes the evaluation of word breaks much more reliable than than the evaluation of phoneme breaks. As mentioned before, the true break between a pair of phonemes can be indeterminate. So it is sometimes illegitimate to specify a break location 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.

We used two metrics 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 $a = t/n$. This tells us how likely it is than an induced break is correct. 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 we compute the accuracy and hit-rate of the induced segmentation over all breaks, including word breaks. Then we also compute the accuracy and hit-rate of the segmentation when considering only the word breaks. That is, we report what the accuracy and hit-rate would be if the answer key contained only the word breaks. Finding word breaks is, in some sense, more important than finding phonemic boundaries, and this is why we perform this additional evaluation. We also report the total number of true breaks and the total number of induced breaks for each experiment.

7 Experimental Results

We have outlined 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.

This algorithm constitutes a very 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: We ran the basic segmentation process described above on both stimulus streams to obtain the induced breaks. We then compared the induced breaks to the true breaks for each stimulus stream. The results are shown in Table 1.

Table 1: Results for Experiment 1.

Key	Dataset	True Breaks	Induced Breaks	Accuracy (Random)	Hit Rate (Random)
All Breaks	Stream A	265	205	0.546 (0.120)	0.411 (0.085)
	Stream B	271	247	0.441 (0.131)	0.387 (0.107)
Word Breaks	Stream A	89	205	0.341 (0.082)	0.764 (0.166)
	Stream B	91	247	0.308 (0.091)	0.791 (0.214)

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 does not drop significantly 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 were correct. Of those 112, 70 of them were at word boundaries, and only 42 were at phoneme boundaries.

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 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 do this, we trained an SOM on the spectrogram data of each stimulus stream to obtain SOM_A and SOM_B. Then we used SOM_A to tokenize the spectrogram data from stimulus stream B and vice versa. Then we trained a *VE* model on each of the token sequences and induced a segmentation. Once again we used the manually labeled true breaks to evaluate the induced segmentation. The results are shown in Table 2.

Table 2: Results for Experiment 2.

Key	Dataset	True Breaks	Induced Breaks	Accuracy (Random)	Hit Rate (Random)
All Breaks	Stream A	265	303	0.290 (0.126)	0.306 (0.127)
	Stream B	271	244	0.279 (0.129)	0.244 (0.105)
Word Breaks	Stream A	89	303	0.195 (0.086)	0.596 (0.250)
	Stream B	91	244	0.184 (0.088)	0.473 (0.209)

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. 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 enough to induce a decent segmentation.

Experiment 3: This experiment is intended to replicate Saffran's experiment on infants. The algorithm learned an audio and language model based on stimulus stream A. Then it was asked to use that model to segment stimulus stream B. This is analogous to an infant learning to segment stimulus stream A by listening to it, and then being played stream B.

We trained an SOM on the spectrogram data of each stimulus stream to obtain SOM_A and SOM_B. Then we used SOM_A to tokenize the spectrogram data from stimulus stream A and from stimulus stream B. We trained a *VE* model using the tokens from stimulus stream A. Finally we used that model to induce a segmentation on the tokens from stimulus stream B. The induced breaks were checked against the true breaks of stream B. We then repeated this experiment by training the audio and language models on stream B, and using them to segment stream A.

Table 3: Results for Experiment 3.

Key	Dataset	True Breaks	Induced Breaks	Accuracy (Random)	Hit Rate (Random)
All Breaks	Stream A	265	18	0.167 (0.127)	0.011 (0.009)
	Stream B	271	6	0.167 (0.163)	0.004 (0.004)
Word Breaks	Stream A	89	18	0.000 (0.087)	0.000 (0.018)
	Stream B	91	6	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 audio model. Instead, the language model trained on one language is insufficient to induce a segmentation in another. This is, of course, exactly as we would expect.

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. The following shows the number of votes at the first 50 vote locations used to segment stimulus stream A in both experiment 1 and experiment 3.

Votes A Exp 1: 0 2 0 3 1 3 2 0 0 0 4 0 7 0 5 1 1 1 1 1 7 0 1 0 1 0 7 0 0 6 0 4 2 1 1 6 0 0 6 2 1 0 3 0 4 7 1 0 0 0
Votes A Exp 3: 0 1 0 1 0 0 3 3 1 1 2 4 2 1 2 1 2 1 2 2 2 2 4 2 2 1 4 1 2 1 1 5 2 1 1 2 6 1 0 1 3 1 1 1 2 3 2 2 4 2

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. The votes are evenly spaced, with few distinguishable "peaks" or "valleys". 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 break the threshold V_t, and 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. Those results are omitted for lack of space.

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.

8 Conclusions and Future Work

We have described an unsupervised technique for transforming spoken audio into a discrete sequence of tokens suitable for segmentation by the Voting Experts algorithm. We have shown that the *VE* model is capable of inducing an accurate segmentation on an audio stimulus stream with very limited training data. Finally, we have shown 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.

The segmentation induced by our algorithm was clearly imperfect. However, there is significant potential for improvement. The method of tokenization is currently fairly simple. Using a GGSOM to cluster instances of a power spectrum produces a very coarse representation of the data. More sophisticated methods of feature extraction exist, and could certainly be put to good use to tokenize the audio stream.

However, this seems to be the wrong way to go. Audio is an intrinsically continuous and real-valued domain. It seems more promising to replace the Voting Experts algorithm with one that uses the same information theoretic markers to directly split the continuous stream. This would require defining the notions of internal and boundary entropy for the continuous domain, which is definitely possible. This certainly seems more appropriate than forcing the data to conform to metrics designed to work on entirely different data.

It is unclear how far these methods can be pushed. Additional research suggests that humans use a lot more than statistical information to learn how to segment spoken language [2]. This includes pauses between words as well as intonation and other prosodic cues. The *VE* model may be able to recognize some of these cues as well, but that is simply speculative. The only solution is to improve the application of the model, and see how well it can perform. Perhaps these ideas can even be combined with more traditional language processing techniques to further improve performance. Either way, it is certainly worthwhile to attempt to discover the principles behind the human audio segmentation behavior. This kind of development is precisely what's needed to produce human-level speech processing software.

Acknowledgments

We would like to thank Peter Wong for developing a system to visualize induced breaks in an audio sequence. We 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. Finally, we would like to thank Paul Cohen from the University of Arizona for generously providing the source code for the original Voting Experts algorithm.

References

[1] Jenny R. Saffran, Richard N. Aslin, and Elissa L. Newport. Statistical learning by 8-month-old infants. *Science*, 274(5294):1926–1928, December 1996.

[2] Jenny R. Saffran, Elizabeth K. Johnson, Richard N. Aslin, and Elissa L. Newport. Statistical learning of tone sequences by human infants and adults. *Cognition*, 1999.

[3] Paul Cohen, Niall Adams, and Brent Heeringa. Voting experts: An unsupervised algorithm for segmenting sequences. *Journal of Intelligent Data Analysis*, 2007.

[4] Matthew Miller and Alexander Stoytchev. Hierarchical voting experts: An unsupervised algorithm for hierarchical sequence segmentation. In *Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL)*, 2008.

[5] Jenny R. Saffran, Elissa L. Newport, Richard N. Aslin, and Rachel A. Tunick. Incidental language learning: Listening (and learning) out of the corner of your ear. *Psychological Science*, 1997.

[6] David M. Magerman and Mitchell P. Marcus. Parsing a natural language using mutual information statistics. In *National Conference on Artificial Intelligence*, pages 984–989, 1990.

[7] A. Kempe. Experiments in unsupervised entropy-based corpus segmentation, 1999.

[8] Margaret A. Hafer and Stephen F. Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10(11-12):371–385, 1974.

[9] Carl de Marcken. The unsupervised acquisition of a lexicon from continuous speech. Technical Report AIM-1558, 1995.

[10] Mathias Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 280–287, 2003.

[11] Michael R. Brent. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 1999.

[12] Claude Shannon. Prediction and the entropy of printed english. Technical report, Bell System Technical Journal, 1951.

[13] Lamere P. Kwok P. Raj B. Gingh R. Gouvea E. et al. Walker, W. Sphinx-1: A flexible open source framework for speech recognition. Technical Report TR-2004-139, Nov 2004.

[14] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. pages 509–521, 1988.

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

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