Unsupervised Segmentation of Audio Speech Using the Voting Experts Algorithm

Matthew Miller **Peter Wong** **Alexander Stoytchev**
Developmental Robotics Lab
Iowa State University
mamille@iastate.edu, pwwong@iastate.edu, 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. 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.

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 (Saffran, Aslin, & Newport 1996). 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 (Saffran, Aslin, & Newport 1996)(Saffran *et al.* 1999). Furthermore, Saffran and others have suggested that humans use statistical properties of the audio stream to induce segmentation. 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. The ability to learn to segment audio speech is useful in and of itself, but also opens up doorways for the exploration of more natural and human-like language learning.

Paul Cohen has suggested an unsupervised algorithm called Voting Experts (*VE*) that uses the information theoretical properties of internal and boundary entropy to segment discrete time series into categorical episodes (Cohen, Adams, & Heeringa 2007). *VE* has previously demonstrated, among other things, the ability to accurately segment plain text that has had the spaces and punctuation removed. 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 and real valued.

Additionally, it is difficult to evaluate an algorithm that tries to find logical breaks in audio streams. In continuous speech, the exact boundary between phonemes or between words is often indeterminate. Furthermore, there are no available audio datasets with all logical breaks labeled, and given an audio stream it is unclear where all the logical breaks even are. What counts as a logical break? It is difficult to quantify the level of granularity with which human beings break an audio stream, and more so to specify the limits of any rational segmentation. This paper describes a method to address these problems.

Our results show that we are able to segment audio sequences with accuracy significantly better than chance. However, given the limitations already described, we are still not at a point to speak to the objective quality of the segmentation.

Related Work

Our work is directly inspired by the psychological studies of audio segmentation in human beings (Saffran, Aslin, & Newport 1996)(Saffran *et al.* 1999)(Saffran *et al.* 1997). These studies show us that the unsupervised segmentation of natural language is possible, and does not require prohibitively long exposure to the audio. 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 (Magerman & Marcus 1990)(Kempe 1999)(Hafer & Weiss 1974)(de Marcken 1995)(Creutz 2003)(Brent 1999)(Ando & Lee 2000). 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.

Gold and Scassellati have created an algorithm specifically to segment audio speech using the MDL model (Gold & Scassellati 2006). They recorded 30 utterances by a single speaker and used MDL techniques to compress the representation of these utterances. They then labeled each compressed utterance as positive or negative, depending on

whether the original utterance contained a target word, and then trained several classifiers on the labeled data. They used these classifiers to classify utterances based on whether they contained the target word. This technique achieved moderate success, but the dataset was small, and it does not produce word boundaries, which is the goal of this work.

This work makes use of the Voting Experts (*VE*) algorithm. *VE* was designed to do with discrete token sequences exactly what we are trying to do with real audio. That is, given a large time series, specify all of the logical breaks so as to segment the series into categorical episodes. The major contribution of this paper lies in transforming an audio signal so that the VE model can be applied to it.

Overview of Voting Experts

The *VE* algorithm is based on the hypothesis that natural breaks in a sequence are usually accompanied by two information theoretic signatures (Cohen, Adams, & Heeringa 2007)(Shannon 1951). 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 the 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* preprocessors the time series to build an n-gram trie, which represents all its possible 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. This algorithm can be run in linear time with respect to the length of the sequence, and can be used to segment very long sequences. For further details, see the journal article (Cohen, Adams, & Heeringa 2007).

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 these 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 use this model to segment audio speech.

The model of segmenting based on low internal entropy and high boundary entropy is also closely related to the work in psychology mentioned above (Saffran *et al.* 1999). 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. So we would like to be able to use VE to segment audio speech, both to test this hypothesis and to possibly facilitate natural language learning.

Experimental Procedure

Our procedure can be broken down into three steps. 1) Temporally discretize the audio sequence while retaining the relevant information. 2) Tokenize the discrete sequence. 3) Apply VE to the tokenized sequence to obtain the logical breaks. These three steps are described in detail below, and illustrated in Figure 2.

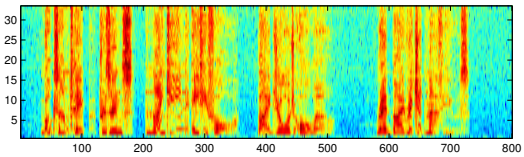

Figure 1: A voiceprint of the first few seconds of one of our audio datasets. The vertical axis represents 33 frequency bins and the horizontal axis represents time. The intensity of each frequency is represented by the color. Each vertical line of pixels then represents a spectrogram calculated over a short Hamming window at a specific point in time.

Step 1

In order to discritize the sequence, we used the discrete Fourier transform in the Sphinx software package to obtain the spectrogram information (Walker *et al.* 2004). We also took advantage of the raised cosine windower and the pre-emphasizer in Sphinx. The audio stream was windowed into 26.6ms wide segments called Hamming windows, taken every 10ms (*i.e.* the windows were overlapping). The windower also applied a transformation on the window to emphasize the central samples and de-emphasize those on the edge. Then the pre-emphasizer normalized the volume across the frequency spectrum. This compensates for the natural attenuation (decrease in intensity) of sound as the frequency is increased.

Finally we used the discrete Fourier Transform to obtain the spectrogram. This is a very standard procedure to obtain the spectrogram information of an audio speech signal, and technical explanation of each of these steps is available in the Sphinx documentation (Walker *et al.* 2004).

We performed the Fourier Transform at 64 points. However, since we are only concerned with the power of the audio signal at each frequency level, and not the phase, then the points are redundant. Only the first 33 contained unique information. This transformation converted a 16kHz mono audio file into a sequence of spectrograms, representing the intensity information in 33 frequency bins, taken every 10ms through the whole file. These spectrograms can be viewed as a voiceprint representing the intensity information over time. Figure 1 shows a voiceprint taken from the beginning of one of the datasets used in our experiments.

Step 2

After discretization the next step is tokenization. Once we obtained the spectrogram of each Hamming window over the entire audio sequence, we converted it to a time series composed of tokens drawn from a relatively small alphabet. In order to do this we trained a Self Organizing Map (SOM) on the spectrogram values (Kohonen 1988).

An SOM can be used as a clustering algorithm for instances in a high dimensional feature space. During training, instances are presented to a 2D layer of nodes. Each node has a location in the input space, and the node closest to the given instance "wins." The winning node and its neighbors are moved slightly closer to the training instance in the input space. This process is repeated for some number of inputs. Once training is complete the nodes in the layer should be organized topologically to represent the instances presented in training. Instances can then be classified or clustered based on the map. Given a new instance, we can calculate the closest node in the map layer, and the instance can be associated with that node. This way we can group all of the instances in a dataset into clusters corresponding to the nodes in the SOM.

However, this approach has its drawbacks. For instance, it requires the specification of a set number of nodes in the network layer before training begins. Layer size selection is not an inconsequential decision. Selecting too many nodes means that similar instances will be mapped to different nodes, and selecting too few means dissimilar instances will be mapped to the same one.

Instead of guessing and checking, we used a Growing Grid self organizing network (Fritzke 1995). The Growing Grid starts with a very small layer of SOM nodes arranged in a rectangular pattern. It trains these nodes on the dataset as usual, and maps the dataset to the nodes based on their trained values. The node whose mapped instances have the highest variance is labeled as the error node. Then a new row or column is inserted into the map between the error node and its most dissimilar neighbor. The new nodes are initialized as the average of the two nodes they separate, and the map is retrained on the entire dataset.

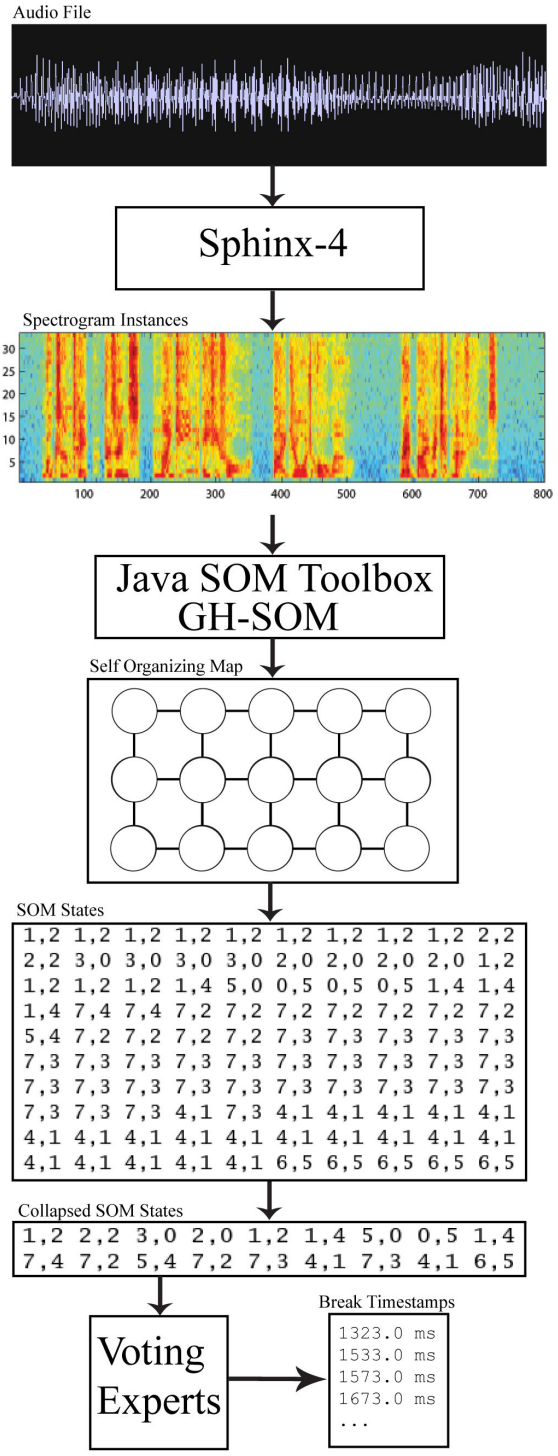

Figure 2: Illustration of the Audio Segmentation Procedure.

The stopping criterion is specified by an error parameter τ. If the variance of the error node is less than τ times the variance of the entire dataset, no new rows or columns are added. This effectively ensures that no single node will account for more than τ of the total error in the dataset. This way the SOM ends up sized appropriately for the particular problem, and the data is spread evenly through the nodes. For our experiments we used a $\tau = 0.01$. We used the implementation of a Growing Hierarchical SOM (GH-SOM) in the Java SOM Toolbox to train our Growing Grid (Dittenbach, Merkl, & Rauber 2000).

After training a Growing Grid SOM on the spectrogram data we used it to classify each instance the dataset. Each node in the SOM was represented by a unique label - its coordinates in the network layer. For instance the node with layer coordinates $(1, 2)$ was represented by the string "1,2". So the clustering produced a sequence of node labels corresponding to each spectrogram instance (see Figure 3). 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 period of silence by a single state, as well as the same sound held for several time steps (see Figure 4). This way our segmentation algorithm only looked at changes between SOM states, and not the duration that the sound stayed the same.

Step 3

In order to segment the tokenized sequence, 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, we produced the time stamps of all of the induced break locations in the audio stream.

Experiments

We performed two experiments to test this algorithm.

Experiment 1 First we used text-to-speech software to generate spoken audio. We modeled our dataset on the audio used for the study of speech segmentation in 8-month-old infants. In that study the infants were played artificially generated speech consisting of four made up words "golabu," "tupiro," "bidaku" and "padoti," in random order. We composed a randomly ordered list of 900 instances of the made up words, and then generated audio speech using the built in text-to-speech synthesizer on a Macintosh laptop. We chose their most natural sounding voice, called "Alex," set on medium speed. This resulted in approximately 10 minutes of speech.

We then ran the segmentation process described above on the audio to obtain the induced breaks. That is, we used Sphinx to obtain the spectrogram information for each Hamming window, trained a Growing Grid SOM on the spectrogram data, and then used the SOM to convert the spectrogram data into a sequence of SOM node labels. Figure 3 shows the first 100 labels in a sample sequence. It is evident that most nodes are repeated several times in a row, and replacing the repeated nodes with a single instance produces the sequence in Figure 4. The VE algorithm was then run on this collapsed sequence, each coordinate pair being used as a fundamental token by the Voting Experts algorithm. VE induced breaks between the coordinate pairs, and by re-expanding the sequence to its original length, we calculated the timestamps of the induced breaks. This entire procedure is visualized in Figure 2.

```
1,2 1,2 1,2 1,2 1,2 1,2 1,2 1,2 1,2 2,2
2,2 3,0 3,0 3,0 3,0 2,0 2,0 2,0 2,0 1,2
1,2 1,2 1,2 1,4 5,0 0,5 0,5 0,5 1,4 1,4
1,4 7,4 7,4 7,2 7,2 7,2 7,2 7,2 7,2 7,2
5,4 7,2 7,2 7,2 7,2 7,3 7,3 7,3 7,3 7,3
7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3
7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3 7,3
7,3 7,3 7,3 4,1 7,3 4,1 4,1 4,1 4,1 4,1
4,1 4,1 4,1 4,1 4,1 4,1 4,1 4,1 4,1 4,1
4,1 4,1 4,1 4,1 4,1 6,5 6,5 6,5 6,5 6,5
```

Figure 3: The coordinates of the SOM nodes corresponding to the first 100 spectrogram instances from a sample artificially generated baby talk audio dataset.

```
1,2 2,2 3,0 2,0 1,2 1,4 5,0 0,5 1,4
7,4 7,2 5,4 7,2 7,3 4,1 7,3 4,1 6,5
```

Figure 4: The coordinates of the SOM nodes corresponding to the first 100 spectrogram instances from a sample artificially generated baby talk audio dataset with the repeated states removed.

Experiment 2 We also performed this exact same procedure on 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 we chose this particular audio book is that the text from 1984 was used in the original segmentation experiments with VE (Cohen, Adams, & Heeringa 2007). This experiment was performed to evaluate the procedure's effectiveness on language spoken by a person, as compared to artificially generated language.

Evaluation Methodology

Evaluating the output of the algorithm proved to be very difficult. In fact, one of the major contributions of this paper is the methodology described here for evaluating the algorithm. In the first experiment, the audio was generated artificially. It would seem simple to determine the duration of each phoneme and word, and compare those time stamps with the induced breaks, however there are two separate problems with this approach.

First of all, the speech generation software does not generate regularly spaced phonemes. It actually constitutes the sequence using diphones, so that the sound and duration of one phoneme is affected by those around it. Furthermore, the sound generally fades between one phoneme and the next, resulting in an ambiguous break location.

Secondly, there exists more than one logical break location between certain phonemes. In particular, there are silent spaces between some words and after certain phonemes. It is acceptable to place a break anywhere in that silence, or even one break at the beginning and one at the end. In other words, there is much more lee-way in proper audio segmentation than one might expect.

These problems are, of course, exacerbated in the case of experiment 2, which uses continuous natural speech instead of the regular artificially generated audio from experiment 1. When evaluating the audio from experiment 1, there are only a limited number of phonemes in use, and a limited number of ways they are combined. This is not the case in experiment 2. Evaluating the breaks then involves solving a different problem at each suggested break location. No two proposed breaks look alike, and so each one is a unique judgment call.

Given these constraints, we tested our segmentation 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. 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 output 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.

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 that an induced break is correct. However, this doesn't tell us how well the algorithm is really doing. After all, for a break to be considered "correct" it simply has 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, we compared the performance of our algorithm with the performance of randomly generated breaks.

Over the 10 minutes of audio in experiment 1, our algorithm induced roughly 4000 breaks. Experiment 2 had roughly four times that many. It would have been impossible to manually check every single one of the 20,000 breaks induced in the two experiments. Instead, we chose a subset of the data, and checked over that. There are several reasons to think that "spot checking" the algorithm is a good measure of its overall accuracy. In experiment 1 the audio stream consists of the same four words repeated over and over in random order. So we would expect the segmentation algorithm to perform fairly uniformly everywhere. In experiment 2 this was not the case, but we strove to sample enough points to ensure that the estimated accuracy was reliable.

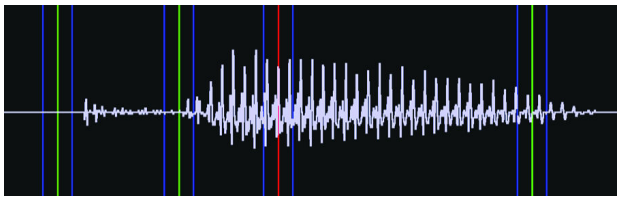

Figure 5: A short sample of the audio from one of our experiments which shows the breaks generated by our algorithm. The "correct" breaks are highlighted in green. The two blue lines around each break show the bounds of the 13ms window.

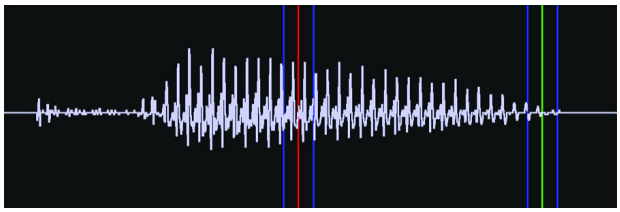

Figure 6: The same audio segment as shown in Figure 5, but with the randomly generated breaks shown instead.

In order to check subsets of the data, 5 sections of 1 minute each were randomly chosen from the audio stream of each experiment. The breaks in these 1 minute segments were recorded. At the same time, for each of these sections we randomly generated the same number of break timestamps over the same 1 minute. 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 5 and 6).

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 we compared the performance of our algorithm to an algorithm that randomly generates roughly the same number of breaks. Figure 5 shows an sample section of audio with the induced breaks drawn in. Figure 6 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.

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 20 files, 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 4 pairs of files to evaluate - 2 pairs taken from each experiment. 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 1. The

agreement between our graders is fairly high, considering the subjective nature of most audio break judgements. Typically, an intercoder reliability of 0.8 is considered acceptable.

Table 1: The Intercoder Reliability for experiments 1 and 2

Experiment	Total Breaks	Agreed Breaks	ICR
Exp 1	1542	1346	0.873
Exp 2	1564	1356	0.867

Results

The graded segmentation results are shown in Table 2. For each experiment the breaks induced by our algorithm are shown next to the breaks that were randomly assigned. As you can see, the VE segmentation performed substantially better than chance. In both experiments the accuracy was above 80%, which is considerably good. However, the probability of a random break being placed at a correct location is above 60% in both datasets. It is impossible to know whether a significant portion of our algorithm's breaks were placed "randomly," (*i.e.* for bad reasons) 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 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 could not perform that experiment at this time.

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 5 and 6 illustrate a typical example of the difference in segmentation.

Table 2: Accuracy Results for Experiments 1 and 2

Experiment	Total Breaks	Correct Breaks	Accuracy
Exp 1 - Algorithm	1922	1584	0.824
Exp 1 - Random	1922	1312	0.683
Exp 2 - Algorithm	1910	1538	0.805
Exp 2 - Random	1910	1220	0.639

Conclusions and Future Work

We have described a technique for transforming spoken audio into a discrete sequence of tokens suitable for segmentation by the Voting Experts algorithm. And we have shown that this technique is clearly capable of inducing logical breaks in audio speech. This is a very significant result and demonstrates that the unsupervised segmentation of audio speech based on the information theoretic model of *VE* is possible. We have tested the algorithm on simple "baby talk" inspired by literature on statistical learning in infants (Saffran, Aslin, & Newport 1996). We have also tested it on large audio dataset of spoken English taken from an audio book. This demonstrates its ability to work on real world audio, as well as its tractability when dealing with large datasets. The segmentation, however, is imperfect. There are several possible avenues of future work that might improve the segmentation accuracy of the algorithm.

For example, we decided to use the spectrogram information calculated at discrete time slices as our base instances. We could have used cepstral information, which has been shown more effective in speech recognition tasks. But the spectrogram is more straightforward and applies to audio other than speech. It is possible in future work to use the cepstral coefficients and their derivatives in place of the spectrograms.

It is also possible that the dimension of the Fourier transform might be increased. The SOM might produce better clustering with a higher or lower τ parameter. Or, there might be a better method altogether for finding boundaries that produce low internal entropy of chunks and high boundary entropy between them. There are many possibilities for improvement and future investigation of this procedure. All that can be said right now is that finding such breaks does produce a somewhat logical segmentation of audio speech. It will be interesting to discover whether truly reliable segmentation can be performed this way, and whether these segments can be used as a basis for human-like language learning.

References

Ando, R., 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*, 241–248.

Brent, M. R. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*.

Cohen, P.; Adams, N.; and Heeringa, B. 2007. Voting experts: An unsupervised algorithm for segmenting sequences. *Journal of Intelligent Data Analysis*.

Creutz, M. 2003. 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*, 280–287.

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. *Neural Networks, 2000. IJCNN 2000* 6:15–19 vol.6.

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

Gold, K., and Scassellati, B. 2006. Audio speech segmentation without language-specific knowledge. In *Proceedings of the 2nd Annual Conference on Human-Robot Interaction (HRI-07)*.

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

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

Kohonen, T. 1988. Self-organized formation of topologically correct feature maps. 509–521.

Magerman, D. M., and Marcus, M. P. 1990. Parsing a natural language using mutual information statistics. In *National Conference on Artificial Intelligence*, 984–989.

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.; 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*.

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*.

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

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.