Hierarchical Voting Experts: An Unsupervised Algorithm for Hierarchical Sequence Segmentation

Matthew Miller and Alexander Stoytchev
Developmental Robotics Lab
Iowa State University
{mamille | alexs}@iastate.edu

***Abstract*—This paper extends the Voting Experts (VE) algorithm for unsupervised segmentation of sequences to create the Hierarchical Voting Experts (HVE) algorithm for unsupervised segmentation of hierarchically structured sequences. The paper evaluates the strengths and weaknesses of the HVE algorithm to identify its proper domain of application. The paper also shows how higher order models of the sequence data can be used to improve lower level segmentation accuracy.**

I. Introduction

The world is too complex to be considered all at once, both computationally and conceptually. Instead, it must be broken into manageable pieces, or *chunks*, and dealt with one piece at a time [1]. However, this is not a trivial task. It isn't clear what segmentation strategy one should use, or even what metric should be used to evaluate the quality of a segmentation.

Human beings have an astounding and apparently innate ability to induce such a segmentation [2], and this mechanism has been variously described and measured [1], [3], [4], [5], [6]. Modeling this process would certainly be an academically and practically fruitful endeavor. The Voting Experts (*VE*) algorithm suggests just such a model, and has demonstrated the capability to accurately segment natural language text [7]. It proposes that chunks have a certain *signature*, *i.e.*, they exhibit two information theoretic characteristics, namely low internal entropy and high boundary entropy. In other words, chunks are composed of elements that are frequently found together, and that are found together in many different circumstances. *VE* looks for these two properties and uses them to segment text. It is surprisingly powerful given its simplicity, suggesting that the principle of segmenting based on low internal entropy and high boundary entropy is promising.

Real world data often exhibits an inherently hierarchical structure, and it is well known that humans chunk the world hierarchically [1], [3]. 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 paper extends the *VE* algorithm to segment hierarchically structured sequences. We show that *VE* can be generalized to work on hierarchical data and investigate the applicability of this extension to determine its strengths and limitations. More specifically, we strive to understand when the underlying information theoretic model for segmentation is valid, and when it is not. We then show that the higher order models can be used to improve the accuracy of the segmentation at lower levels.

II. Related Work

Several algorithms have been described in the literature for unsupervised sequence segmentation. In particular there exist segmentation algorithms that use statistical properties of sequences [8], [9], [10], [11], [12]. There also exist models of infant speech segmentation based on clustering or Bayesean approaches [13], [14]. Additionally the SEQUITUR algorithm has demonstrated the ability to discover hierarchical structure in sequence data, and has been altered to perform unsupervised segmentation tasks [15], [7]. However, its segmentation performance is inferior to that of *VE* [7].

The work presented here, however, is more closely related to the field of Statistical Learning. A paper by Saffran, Johnson, Aslin and Newport demonstrated that humans possess a general mechanism for segmenting audio data [5]. They claim that the segmentation was induced based on "statistical cues." These are the "sequential properties" of the phonemes or tones [5]. Specifically, given two sequential tones A and B, the probability that B follows A is generally higher if the two tones are part of the same word, and generally lower if there is a word break between them. The study concludes that humans must use these statistical cues to segment audio streams [5]. But these cues are simply more impoverished versions of the "low internal entropy" and "high boundary entropy" signatures of chunks used by *VE*.

When we say that a sequence has low internal entropy, this literally means that the transition probability between each element in the sequence is high. When we say that a sequence has high boundary entropy we literally mean that, given the sequence, there is no particular element that has a high probability of being next. Specifying these markers in terms of information theory [16] gives us a very clear and well understood characterization. The *VE* model can be seen as a refinement and artificial implementation of this model of human segmentation. This model may or may not capture the true human strategy. However, it seems complimentary to the findings of Saffran, Aslin and others.

III. Voting Experts

The *VE* model uses two measures of entropy 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 entropy of a subsequence $c = (s_j, \ldots, s_k)$ is given by $H_I(c) = log(Pr(c))$. 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 $P(h,c) = Pr(s_{k+1} = e_h | s_j, \ldots, s_k)$.

VE uses a sliding window to make local splitting decisions. This allows the algorithm to run in linear time, and work on very large datasets. The *VE* algorithm consists of three main steps. Given a sequence of characters for segmentation:

Step 1*:* Build an ngram trie of the sequence and use it to calculate the internal entropy and boundary entropy of each subsequence of length less than or equal to n. Each value is then standardized across all subsequences of the same length. Let H_I^l be the average internal entropy for all sequences of length l, and σ_l be the standard deviation of the internal entropy for all sequences of length l. Then the standardized internal entropy of a chunk c of length l is defined as $\hat{H}_I(c) = (H_I(c) - H_I^l)/\sigma_l$. The boundary entropy is standardized analogously.

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 technically, 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. Both experts use the trie to perform these calculations.[1]

A window length of $N = 7$ was chosen for the experiments in this paper. This was the window size used to test the original *VE* algorithm, and it is appropriate given the average length of English words.

Step 3*:* Choose a threshhold 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 t.

For further technical and implementation details of the algorithm, or for a comparison of *VE* with other segmentation algorithms, see the journal article [7].

IV. 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 ngram 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.

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 ngram 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.

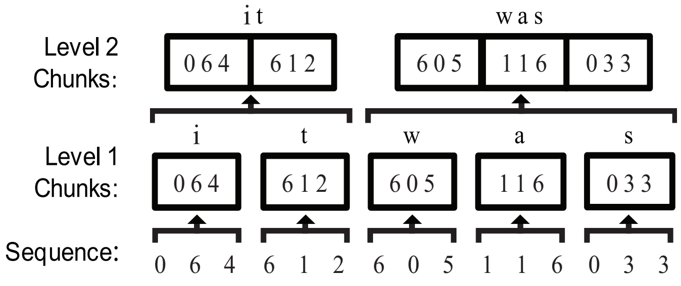

Fig. 1. An illustration of hierarchical chunking. The digits are grouped into chunks that represent letters. Those chunks are then grouped into chunks that represent words.

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 1). 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, we performed this experiment, 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

[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 [7]. 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.

a Fourier transform of an audio signal. It is one step further towards 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.

V. Experiments with HVE

We designed several experiments 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 entropy, 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, 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.

Dataset: For our experiments we used the first 35,000 words (converted to lower case and stripped of punctuation and spaces) of George Orwell's novel "1984" as our base text. This was one of the benchmark datasets used to evaluate the original *VE* algorithm, so it was chosen for comparison [7]. To perform our experiments, we translated this data 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 [7]. However, the accuracy of the segmentation asymptotically approaches an upper bound, and there is little utility in using a corpus much larger than our base dataset. Additionally, our 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.

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 [7]. 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 threshhold t in step three of the *VE* algorithm has an effect on these metrics. Specifically, as t is raised, it

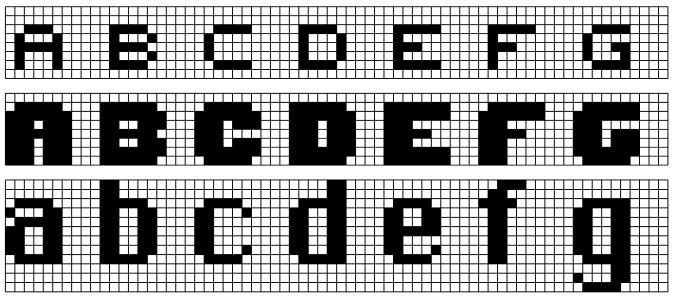

Fig. 2. Fonts 1, 2 and 3 used in Experiment 1. Fonts 1 and 2 both have resolution 8x8 for each letter, and font 3 has resolution 12x8. However, even though fonts 1 and 2 have the same resolution, font 2 is more complex than font 1, in that each letter is composed of more unique pixel columns.

takes more votes to induce a split. This causes the accuracy of the segmentation to go up, and the hit rate to go down. Conversly, lowering t generally lowers the accuracy and raises the hit rate. In each of our experiments we chose the t that struck a balance between the two. This method was also used in the original *VE* algorithm [7]. We hope to eventually find a principled way to set t, so that the algorithm needs no hand-tuning.

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 3). We used three fonts, whose resolution and complexity varied (see Figure 2). For each font we transcribed the text and then ran a two-level *HVE* 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 I).

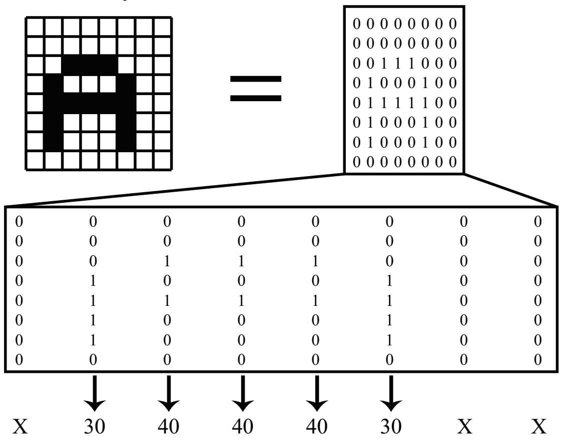

Fig. 3. An illustration of the conversion of the letter "a" to a sequence of integers corresponding its pixel columns. The pixels are converted to bits, and each column of bits is converted to a decimal number. The white space is removed before and after each letter so that there are no boundary markers in the sequence.

TABLE I
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

A. Strengths and Limitations of HVE

In addition to demonstrating its power, we also set out to find the limitations of *HVE*, 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 solve. Table II illustrates the translation applied to the dataset for each of these experiments. The results show that *HVE* had poor performance in experiments 2 and 3, but was successful on experiment 4 (see Table III).

TABLE II
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: We translated each character in the text into its Morse code representation. We represented the Morse code with a sequence of bits - 0 corresponded to “dit” and 1 corresponded to “dah.” Then we ran a two-level *HVE* on the data to attempt to segment the Morse code into letters, and then the letters into words.

Experiment 3: We translated each character in the text into its standard ASCII octal representation. So each character was translated into 3 digits ranging from 0 to 7. We then ran a two-level *HVE* on the translated data.

Experiment 4: We generated a mapping from each letter to a random three digit octal number. We used this mapping to translate the base text by replacing each character in it with its corresponding three digits. We then ran two-level *HVE* on this data to segment the letters and then the words.

TABLE III
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

B. Phoneme Segmentation

One of the main questions we aim to eventually answer is whether an algorithm like *HVE* can be used to chunk in more complex domains like audio speech. Accordingly, text segmentation may not be fully representative of the power of the model. 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. This is not true in practice because of the difficulty of dealing with audio data. However, we designed an experiment to test this hypothesis without dealing with audio.

Experiment 5: We used the CMU Pronouncing Dictionary to translate each word in our dataset into its phonemic representation. The CMU Dictionary uses a text base representation of 39 phonemes to represent over 125,000 words. We looked up the phonemic representation of each word in our dataset and replaced it with the corresponding phonemes. 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. We then ran one-level *HVE* on the translated data. Table IV summarizes the results which are analyzed in the next section.

TABLE IV
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

VI. 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. Our 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 entropy" 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 entropy 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 III).

In experiment 5 the segmentation of the phonemes is slightly better than the baseline segmentation of the original text. This is and encouraging result, which seems to indicate that it might be possible to segment audio speech using our model. However, it is impossible to determine from this experiment whether the information theoretic signatures of chunks are more pronounced in audio speech than in written text. Our translation from text to phonemes ignores interactions between words, and is therefore not an entirely complete model. All we can say for sure at this time is that HVE segments single word phonemic representations more accurately than text. It is an interesting open question whether this will hold true for audio speech.

VII. HVE-3E

In addition to extending *VE* to work on hierarchical data, we devised a mechanism to use higher order model information to increase segmentation accuracy at the lower level. To do this we ran standard *HVE* on a given sequence to obtain a sequence of chunks. Then we built the second order model (ngram trie) using those chunks. Finally, we re-ran the algorithm on the original sequence with the addition of a third voting expert. We call this modification of the algorithm *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 4). 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 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.

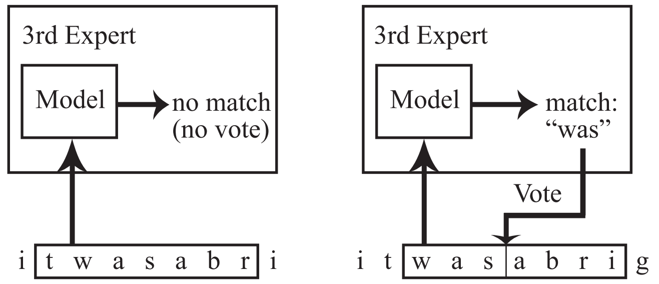

Fig. 4. An illustration of the third voting expert. Given a sliding window, it tries to find a sequence that its model recognizes, starting at the beginning of the window. If it doesn't, it does not vote. If it does, it votes to place a break after that sequence. This vote is combined with those from 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 we re-ran experiments 1 and 4 with the additional expert added. We used the third expert to increase segmentation accuracy on both the first and second level of the hierarchy. Table V demonstrates the improvement at both levels.

TABLE V
EXPERIMENT 6: *HVE-3E*. COMPARE WITH TABLE I AND TABLE III. 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

A. 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 ours [17]. Their algorithm, however, was not hierarchical, and was 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.

VIII. Conclusions and Future Work

We have described a natural extension of the Voting Experts (*VE*) algorithm [7], called Hierarchical Voting Experts (*HVE*), which segments hierarchically structured sequences. We have shown that *HVE* can successfully perform hierarchical segmentation on a variety of datasets. Also, we have 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. We have also demonstrated a technique for improving the segmentation accuracy by making use of higher order models, and shown that it is 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. We intend to explore its applicability to audio speech in particular. But there is no reason to stop there. 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.

Acknowledgments

We would like to acknowledge Paul Cohen and Wesley Kerr from the University of Southern California for generously sending us the source code for the original VE algorithm, and Paul Cohen for his helpful comments about our algorithm.

References

[1] G. Miller, “The magical number seven, plus or minus two: Some limits on our capacity for processing information,” *Psychological Review*, vol. 63, pp. 81–97, 1956.

[2] J. Saffran, R. Aslin, and E. Newport, “Statistical learning by 8-month-old infants,” *Science*, vol. 274, no. 5294, pp. 1926–1928, 1996.

[3] J. Fiser and R. Aslin, “Encoding multielement scenes: statistical learning of visual feature hierarchies.” *J Exp Psychol Gen*, vol. 134, no. 4, pp. 521–537, November 2005.

[4] ——, “Statistical learning of higher-order temporal structure from visual shape sequences,” *Journal of Experimental Psychology*, 2002.

[5] J. Saffran, E. Johnson, R. Aslin, and E. Newport, “Statistical learning of tone sequences by human infants and adults,” *Cognition*, 1999.

[6] J. Fiser and R. Aslin, “Unsupervised statistical learning of higher-order spatial structures from visual scenes,” *Psychological Science*, 2001.

[7] P. Cohen, N. Adams, and B. Heeringa, “Voting experts: An unsupervised algorithm for segmenting sequences,” *Intelligent Data Analysis*, 2007.

[8] M. Brent, “An efficient, probabilistically sound algorithm for segmentation and word discovery,” *Machine Learning*, 1999.

[9] M. Creutz, “Unsupervised segmentation of words using prior distributions of morph length and frequency,” in *Proceedings of the 41st Annual Meeting on Associ. for Computational Linguistics*, 2003, pp. 280–287.

[10] M. Hafer and S. F. Weiss, “Word segmentation by letter successor varieties,” *Information Storage and Retrieval*, vol. 10, no. 11-12, pp. 371–385, 1974.

[11] A. Kempe, “Experiments in unsupervised entropy-based corpus segmentation,” 1999.

[12] D. Magerman and M. Marcus, “Parsing a natural language using mutual information statistics,” in *National Conference on Artificial Intelligence*, 1990, pp. 984–989.

[13] D. Swingley, “Statistical clustering and the contents of the infant vocabulary,” *Cognitive Psychology*, 2005.

[14] S. Goldwater, T. Griffiths, and M. Johnson, “Contextual dependencies in unsupervised word segmentation,” in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, Morristown, NJ, 2006, pp. 673–680.

[15] C. Nevill-Manning and I. Witten, “Identifying hierarchical structure in sequences: A linear-time algorithm,” 1997.

[16] C. Shannon, “Prediction and the entropy of printed english,” Bell System Technical Journal, Tech. Rep., 1951.

[17] J. Cheng and M. Mitzenmacher, “The markov expert for finding episodes in time series,” in *Proceedings of the Data Compression Conference*, 2005, pp. 454–454.