Voting Experts: An Unsupervised Algorithm for Segmenting Sequences

Paul Cohen[1], Niall Adams[2], Brent Heeringa[3]

[1]USC Information Sciences Institute
[2]Department of Mathematics, Imperial College London, UK
[3]Department of Computer Science, University of Massachusetts

July 15, 2006

Abstract

We describe a statistical signature of *chunks* and an algorithm for finding chunks. While there is no formal definition of chunks, they may be reliably identified as configurations with low internal entropy or unpredictability and high entropy at their boundaries. We show that the log frequency of a chunk is a measure of its internal entropy. The VOTING-EXPERTS exploits the signature of chunks to find word boundaries in text from four languages and episode boundaries in the activities of a mobile robot.

1 Introduction

"I have fallen into the custom of distinguishing between *bits* of information and *chunks* of information. ... The span of immediate memory seems to be almost independent of the number of bits per chunk, at least over the range that has been examined to date.

The contrast of the terms bit and chunk also serves to highlight the fact that we are not very definite about what constitutes a chunk of information. For example, the memory span of five words ... might just as appropriately have been called a memory span of 15 phonemes, since each word had about three phonemes in it. Intuitively, it is clear that the subjects were recalling five words, not 15 phonemes, but the logical distinction is not immediately apparent. We are dealing here with a process of organizing or grouping the input into familiar units or chunks, and a great deal of learning has gone into the formation of these familiar units. " —George Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" [25]

So began the story of chunking, one of the most useful and least understood phenomena in human cognition. Although chunks are "what short term memory can hold five of," they appear to be incommensurate in most other respects. Miller himself was perplexed because the information content of chunks is so different. A telephone number, which may be two or three chunks long, is very different from a chessboard, which may also contain just a few chunks but is vastly more complex. Chunks contain other chunks, further obscuring their information content. The psychological literature describes chunking in many experimental situations (mostly having to do with long-term memory) but it says nothing about the intrinsic, mathematical properties of chunks. The cognitive science literature discusses algorithms for forming chunks, but, again, there is little said about what chunks have in common.

Miller was close to the mark when he compared bits with chunks. Chunks may be identified by an information theoretic *signature*. Although chunks may contain vastly different amounts of Shannon information, they have one thing in common: Entropy within a chunk is relatively low, entropy at chunk boundaries is relatively high.

One sees many hints of this signature in the literature on segmentation. Many algorithms for segmenting speech and text rely on some kind of information-theoretic indicator of segment boundaries (e.g., [2, 21, 23, 33, 34, 5]). Edge-detectors in computer vision may be described in similar terms (e.g., [29]). Intriguingly, while common compression algorithms rely on a version of the "low entropy within a chunk" rule, the prediction by partial match (PPM) method, which is apparently superior to other compression algorithms, also attends to the entropy between chunks. One wonders whether the human perceptual and cognitive systems have evolved to detect this "low entropy within, high entropy between" signature. Saffran, Aslin and Newport [28] found something like it in their experiments with speech segmentation by very young infants, and Hauser and his colleages suggest the same mechanism is at work in cotton-top tamarin monkeys [17, 11].

More comically, have you ever wondered why young children freeze in doorways and at the bottom of escalators, or why some people in crowded airline terminals walk out of the jetway and stop, causing a traffic jam behind them? These are examples of transitions from familiar to unfamiliar surroundings, of low to high local entropy. Some people can't handle it.

This paper introduces an unsupervised algorithm called VOTING EXPERTS for finding chunks in sequences. As such, it has applications to segmentation, a task it performs very well. Most of the empirical evaluations of VOTING EXPERTS are on segmentation tasks, because these tasks provide opportunities to compare our algorithm with others. However, segmentation is not our primary interest. If it were, we would focus on supervised segmentation (in which algorithms are trained with many examples of correct segmentation) because it performs better than unsupervised method. Even among unsupervised segmenters, some may perform better than VOTING EXPERTS, and we may be sure that somewhere in the sophisticated arsenal of probabilistic algorithms trained on corpora of millions of words are some that will beat our simple method. So segmentation is not our focus. Rather, our point is that remarkably accurate segmentation may be accomplished by a simple algorithm with very modest data requirements because we have identified a signature of chunks — low entropy within, high entropy between — and designed the algorithm to look for it.

This paper is organized as follows: Section 2 introduces the statistical signature of chunks. Section 3 describes how VOTING EXPERTS works, and Sections 4, 4.3, and 4.4 demonstrates how well it works, including comparisons with the SEQUITUR segmentation algorithm, tests on corpora from several languages, and tests with time series data from a mobile robot. Section 5 describes some related work in the cognitive sciences and computer science.

2 What makes a chunk?

If one strips the spaces from between words and the text to replace *a* with *b*, and *b* with *c*, and so on, the result is quite unrecognizable:

```
tpcfhbouiftupszpgdivoljohpofpguifnptuvtfgvmboemfbtuvoefstuppeq
ifopnfobjoivnbodphojujpobmuipvhidivoltbsfxibutipsuufsnnfnpszdboip
megjwfpguifzbqqfbsupcfjodpnnfotvsbufjonptupuifssftqfdutnjmmfs
ijntfmgxbtqfsqmfyfecfdbvtfuifjogpsnbujpodpoufoupgdivoltjttpejggf
sfoubufmfqipofovncfsxijdinbzcfuxppsuisffdivoltm
```

This text is, in fact, sixty-two words from the opening paragraph of this paper, but because you cannot recognize the words, it is difficult to know which subsequences *are* words. This is how the problem appears to VOTING EXPERTS or, indeed, any unsupervised segmentation problem. The letters have no meaning to VOTING EXPERTS, it does not have a dictionary of words stored somewhere, and it does not know what the text is about. It has nothing to help it find chunks besides the statistical properties of the text, itself. Remarkably, these properties take it a long way.

If you stare at the text long enough, you will see some regularities. The pattern `uif` appears five times, `div` four. Moreover, whenever you see `div` the next two letters are `ol`. In fact, `divol` appears four times in the text and it is the most common substring of three or more letters except `uif` and, of course, its substrings `div`, `ivo`, and `vol`. So which is the chunk, `divol`, or `div`, `ivo`, `vol`, `divo`, or `ivol`? Or perhaps it is a smaller substring, such as `d` or `iv`, or a larger one, such as `divolt`, which appears three times in the text. When the VOTING EXPERTS algorithm is given the text, above, and nothing else, it decides to place a boundary after `divol` once and `divolt` three times. By now you have guessed that `uif` corresponds with the word "the" and `divol` with "chunk." When the text is translated back to English, this is how the algorithm performed:

```
sobe ⋆ gan ⋆ the ⋆ sto ⋆ ry ⋆ of ⋆ chunk ⋆ ing ⋆ one ⋆ ofthe ⋆ most ⋆
use ⋆ fulan ⋆ dle ⋆ astund ⋆ er ⋆ sto ⋆ odphen ⋆ omen ⋆ ain ⋆ huma ⋆
nco ⋆ gnition ⋆ altho ⋆ ugh ⋆ chunks ⋆ are ⋆ wh ⋆ atshort ⋆ ter ⋆ mme ⋆
mo ⋆ rycan ⋆ ho ⋆ ldfive ⋆ ofthe ⋆ yappea ⋆ rtobe ⋆ in ⋆ co ⋆ mmen ⋆
surate ⋆ in ⋆ mosto ⋆ ther ⋆ re ⋆ spe ⋆ ctsm ⋆ iller ⋆ him ⋆ se ⋆
lfwasper ⋆ ple ⋆ xedbe ⋆ cause ⋆ the ⋆ in ⋆ for ⋆ ma ⋆ tion ⋆
co ⋆ nt ⋆ ent ⋆ of ⋆ chunks ⋆ is ⋆ s ⋆ odiff ⋆ er ⋆ ent ⋆ ate ⋆ leph ⋆ on
⋆ en ⋆ um ⋆ ber ⋆ whi ⋆ ch ⋆ maybe ⋆ twoor ⋆ th ⋆ ree ⋆ chunks ⋆ l
```

Hardly perfect segmentation, but consider what the algorithm did right: It found 57% of the word boundaries in the text, and 78% of the boundaries it found were true boundaries. It found several true words, including "the" and "chunks." In fact, it found 19% of the words in the text and it found one or the other boundary of an additional 50% of the words. Some things it found are true words, although it did not get credit for them; for example, "him" is a true word, and was discovered by the algorithm, although the correct word to identify in the text is "himself". Although we will show much better performance from VOTING EXPERTS later on, it is worth showing what the algorithm can do with just

310 characters of text: no spaces, no training instances of good and bad segmentation, no models, no dictionary, nothing but the text.

How does it work? Let us examine some words it found reliably, `the`, `chunk` and `chunks`. Of the 738 subsequences in the text whose length is between one and five characters, these three words are distinctive: They are frequent (relative to their lengths) and the character that follows them in the text is relatively uncertain. `the` is the most frequent three-letter sequence, and `chunk` and `hunks` are the most frequent five-letter sequences. If one ranks sequences by the uncertainty of the character that follows them, then `the` ranks highest, followed by `unks`, `chunks`, and `hunks`. The rank of `chunk` is 24; somewhat lower, but in the top 5% of the 738 subsequences. VOTING EXPERTS attends to the frequency of a sequence and the uncertainty of the character that follows a sequence. It slides a window across the text and votes to place boundaries in such a way as to maximize the frequency of the subsequences between the boundaries, and to maximize the uncertainty of the characters immediately after boundaries. Then it cuts the text at locations that have locally maximum numbers of votes. Apart from adjustments to standardize frequencies across sequences of different lengths, this is a pretty complete description of VOTING EXPERTS.

The important question about VOTING EXPERTS is not how it works but why it works. VOTING EXPERTS is designed to detect two characteristics of chunks: The entropy or unpredictability of elements within a chunk is relatively low, whereas the entropy or unpredictability of elements between chunks is relatively high. By maximizing the frequencies of the chunks it finds, the algorithm minimizes the unpredictability within chunks, and by maximizing the unpredictability of the letters that follow chunks, the algorithm seeks to maximize the unpredictability between chunks.

The equation of within-chunk predictability with chunk frequency is perhaps not obvious. Let $x_1, \ldots, x_n$ be a sequence of length n. The frequency of this sequence in a corpus is just the corpus size times the probability of the sequence, $Pr(x_1, \ldots, x_n)$. By the chain rule for probability we can expand $Pr(x_1, \ldots, x_n)$ into a product of conditional probabilities:

$$Pr(x_1, \ldots, x_n) = Pr(x_1)Pr(x_2|x_1) \ldots Pr(x_n|x_1, \ldots, x_{n-1}) \qquad (1)$$

One notices immediately that the frequency of a chunk will not be very high if any of these conditional probabilities is very low. Apparently the frequency of a sequence $x_1, \ldots, x_n$ has something to do with the predictability of each $x_{i \le n}$ in the sequence given the preceding subsequences $x_1, ..., x_{i-1}$.

A similar argument follows from the definition of the entropy of a random variable X: $H(X) = -\sum_{x \in X} Pr(x) \log Pr(x)$. The entropy is the average *surprisal*, $\log Pr(x)$, associated with variable X taking on value x. The lower the probability that $X = x$, the higher the surprisal. When observing items in a sequence one can condition entropy on items one has already seen:

$$H(x_n|x_1, \ldots, x_{n-1}) = -\sum_{x_n \in X} Pr(x_n|x_1, \ldots, x_{n-1}) \log Pr(x_n|x_1, \ldots, x_{n-1}) \qquad (2)$$

This *conditional entropy* is the average surprisal associated with seeing $X = x$ after having seen $x_1, \ldots, x_{n-1}$. Taking logs of Equation 1 expresses the probability of a sequence in terms of surprisals:

$$\log Pr(x_1, \ldots, x_n) = \log Pr(x_1) + \log Pr(x_2|x_1) + \ldots + \log Pr(x_n|x_1, \ldots, x_{n-1}) \quad (3)$$

Each term on the right hand side of this equation is a surprisal. The first term, $\log Pr(x_1)$ is the unconditional surprisal associated with seeing the first item in the sequence; the second term is the surprisal associated with seeing the second item given the first; and so on. Evidently, the log probability of a sequence, $\log Pr(x_1, \ldots, x_n)$ is the sum of the surprisals associated with each successive item in the sequence. In other words, the log probability of a sequence is just the total surprisal associated with seeing the items in the sequence one at a time. Low surprisals add up to high log probability.

Thus, the log frequency of a chunk in a corpus is just the sum of the surprisals associated with seeing its constituents one at a time. The more frequent a chunk is, the lower the sum of surprisals it has. Voting Experts uses frequency as its indicator of within-chunk predictability.

The second indicator of chunks is high between-chunk entropy. We call it *boundary entropy*, to remind us that it is the entropy of the item that follows a chunk, but this is merely another name for conditional entropy, as shown in Equation 2. When $H(x_n|x_1, \ldots, x_{n-1})$ is high, it indicates that x_{n-1} is the end of a chunk and x_n is the beginning of a new one.

3 The Voting Experts Algorithm

The Voting Experts algorithm takes as input an unsegmented sequence, such as text from which spaces between words have been removed. It calculates statistics of subsequences, or ngrams, with which it decides where to segment the sequence. The algorithm has three phases:

1. Build an ngram trie and calculate standardized frequencies and boundary entropies for each ngram;

2. Calculate a segmentation score for each location in the sequence;

3. Select the locations at which to segment the sequence.

We discuss these in turn.

3.1 The ngram trie and standardized scores

An ngram trie of depth $n+1$ represents all the subsequences of length n of a sequence. For example, $a\ b\ c\ a\ b\ d$ produces the depth 3 trie in Figure 1. Every ngram of length 2 or less in the sequence $a\ b\ c\ a\ b\ d$ is represented by a node in this tree. The numbers in the lower half of the nodes represent the frequencies of the subsequences. For example, the subsequence ab occurs twice, and every occurrence of a is followed by b. The subsequences $b\ c$ and $b\ d$ each occur once, so the frequency of the subsequence b is 2. The children of a node are the extensions of the ngram represented by the node.

The *boundary entropy* of an ngram is the entropy of the distribution of tokens that can extend the ngram. A node n_i has frequency f_i. The children of n_i, denoted $n_{i,1}, ..., n_{i,m}$,

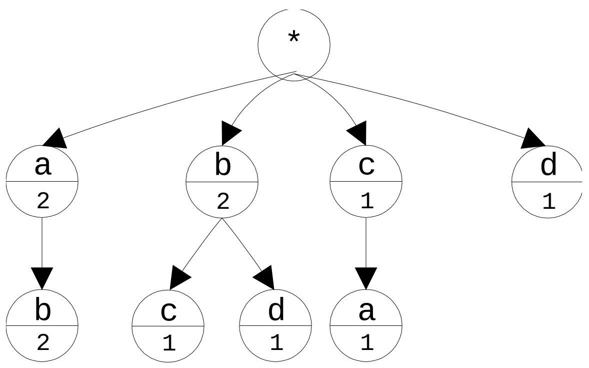

Figure 1: A trie of ngrams of length 2 for the sequence $a\ b\ c\ a\ b\ d$

have frequencies $f_{i,1}, ..., f_{i,m}$. The conditional probability of a node is its frequency divided by the frequency of its parent: $Pr(n_{ij}) = f_{ij}/f_i$. The boundary entropy of a node n_i is

$$H_b(n_i) = -\sum_{j=1}^{m} Pr(n_{ij}) \log Pr(n_{ij})$$

For example, the node a in Figure 1 has entropy equal to zero because it has only one child, $a\ b$, whereas the entropy of node b is 1.0 because it has two equiprobable children, $b\ c$ and $b\ d$. Clearly, only the first n levels of the ngram tree of depth $n+1$ can have node entropy scores.

In most domains, there is a systematic relationship between the length and frequency of patterns: Short patterns are more common than long ones. For instance, in the first 10,000 characters of George Orwell's classic *1984*, 64 of the 100 most frequent patterns are of length 2; 23 are of length 3, and so on. Voting Experts will compare the frequencies and boundary entropies of ngrams of different lengths, but in all cases it will be comparing how *unusual* these frequencies and entropies are, relative to other ngrams of the same length. To illustrate, consider frequencies of the words "a" and "an" in the first 10,000 characters of 1984: "a" occurs 743 times, "an" 124 times, but "a" occurs only a little more frequently than other one-letter ngrams, whereas "an" occurs much more often than other two-letter ngrams. In this sense, "a" is ordinary, "an" is unusual. Although "a" is much more common than "an" it is much less unusual relative to other ngrams of the same length.

To capture this notion of unusualness, we *standardize* the frequencies and boundary entropies of the ngrams. To standardize a sample one subtracts the sample mean and divides by the sample standard deviation: $z_i = (x_i - \bar{x})/s$. This has the effect of expressing each value as the number of standard deviations it is away from the mean. Standardized, the frequency of "a" is 1.3, whereas the frequency of "an" is 3.7. In other words, the frequency of "an" is 3.7 standard deviations above the mean frequency for sequences of the same length. We standardize boundary entropies in the same way, and for the same reason.

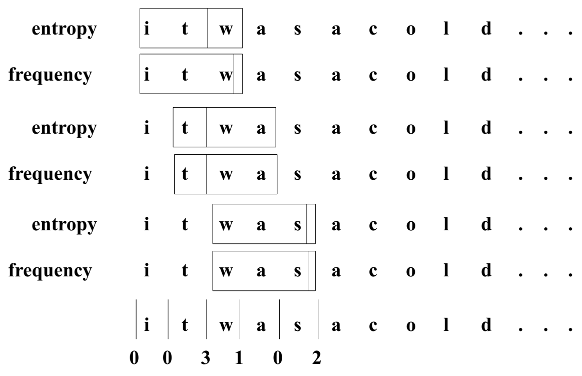

Figure 2: Sample execution of Voting Experts.

3.2 Segmentation scores

In a sequence of length k there are $k-1$ places to draw boundaries between segments, and, thus, there are 2^{k-1} ways to divide the sequence into subsequences. Voting Experts is greedy in the sense that it considers just $k-1$, not 2^{k-1}, ways to divide the sequence. It considers each possible boundary in order, starting at the beginning of the sequence. The algorithm passes a window of length n over the sequence, halting at each possible boundary. All of the locations within the window are considered, and each gets at most one vote from each expert. Because we have two experts, for boundary entropy and frequency respectively, each location may accrue a maximum of $2n$ votes.

The process is illustrated in Figure 2. A window of length 3 is passed along the sequence `itwasacold`. Initially, the window covers `itw`. The entropy and frequency experts each decide where they could best split the sequence within the window (more on this, below). The entropy expert favors a boundary between `t` and `w`, while the frequency expert prefers to cut the sequence between `w` and whatever comes next. Then the window moves one location to the right and the process repeats. This time, both experts decide to place a boundary between `t` and `w`. The window moves again and both experts decide to place the boundary after `s`, the last token in the window.

Note that each potential boundary location (e.g., between `t` and `w`) is seen n times for a window of size n, but it is considered in a slightly different context each time the window moves. The first time the experts consider the boundary between `w` and `a`, they are looking at the window `itw`, and the last time, they are looking at `was`. In this way, each boundary gets up to $2n$ votes, or $n = 3$ votes from each of two experts. The `wa` boundary gets one vote, the `tw` boundary, three votes, and the `sa` boundary, two votes.

The experts use different methods to evaluate boundaries and assign votes. Consider the window `itw` from the viewpoint of the boundary entropy expert. Each location in the window bounds an ngram to the left of the location; the ngrams are `i`, `it`, and `itw`, respectively. Each ngram has a standardized boundary entropy. The boundary entropy expert votes for the location that produces the ngram with the highest standardized boundary entropy. As it happens, for the ngram tree produced from Orwell's text, the standardized boundary entropies for `i`, `it`, and `itw` are 0.2, 1.39 and 0.02, so the boundary entropy expert opts to put a boundary after the ngram `it`.

The frequency expert places a boundary so as to maximize the sum of the standardized frequencies of the ngrams to the left and the right of the boundary. Consider the window `itw` again. If the boundary is placed after `i`, then (for Orwell's text) the standardized frequencies of `i` and `tw` sum to 1.73; if the boundary is placed after `it`, then the standardized frequencies of `it` and `w` sum to 2.9; finally, if it is placed after `itw`, the algorithm has only the standardized frequency of `itw` to work with; it is 4.0. Thus, the frequency expert opts to put a boundary after `itw`.

3.3 Segment the sequence

Each potential boundary in a sequence accrues votes, as described above, and now Voting Experts must evaluate the boundaries and decide where to segment the sequence. The method is a familiar "zero crossing" rule: If a potential boundary has a locally maximum number of votes, split the sequence at that boundary. In the example above, this rule causes the sequence `itwasacold` to be split after `it` and `was`. One problem with the zero crossing rule is that a single vote might be the locally maximum number of votes (if neighboring locations garnered zero votes), causing Voting Experts to split too frequently. We have found empirically that the more votes a location gets, the more likely it is to be a true boundary. Yet the zero crossing rule splits on one vote with the same confidence as it splits on ten votes if both numbers are local maxima. To counteract the tendency to split on small numbers of votes, we do not allow Voting Experts to split unless the number of votes exceeds a minimum number.

4 Evaluation

This section describes performance metrics and experiments with Voting Experts.

4.1 Metrics

The basic experimental question about Voting Experts is whether it cuts sequences where it should. We will call the elements of a sequence *letters*, and subsequences, *words*. The words found by Voting Experts are called *induced words*, to distinguish them from the *true words* it is supposed to find.

The boundaries of induced words stand in six relations to those of true words. These relationships are illustrated graphically in Figure 3, following the convention that horizontal lines denote true words, and vertical lines denote induced words.

1. The induced word boundaries correspond with true word boundaries;
2. Exactly one induced word boundary corresponds with a true word boundary.
3. Both induced word boundaries fall between the true word boundaries.
4. Two or more consecutive induced words "tile" a true word, so that the leftmost boundary of the first induced word is the beginning of the true word, and the rightmost boundary of the last induced word is the end of the true word.
5. Like case 4, except that induced word boundaries correspond to the beginning or the end of the true word but not to both.

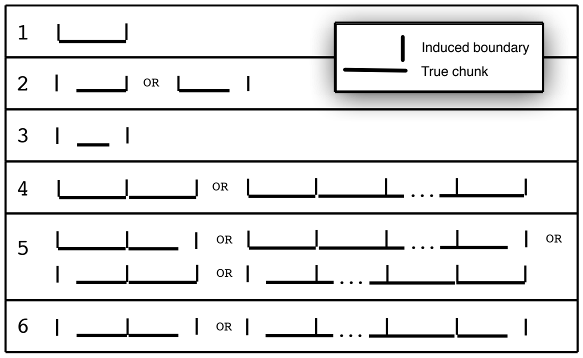

Figure 3: A graphical depiction of the relationships between induced boundaries and true chunk boundaries.

6. No induced word boundary corresponds with either the beginning or the end of the true word.

The cases can be divided into three groups. In cases 1 and 4, induced word boundaries correspond to both ends of true words; in cases 2 and 5, they correspond to one end of the true word; and in cases 3 and 6, they correspond to neither end. We call these cases *exact*, *dangling*, and *lost* to evoke the idea of true words located exactly, dangling from a single induced boundary, or lost in the region between induced boundaries.

Hits and false positives are defined in the usual way: If Voting Experts cuts the word *washington* into the words $*wash * ing * ton*$, it gets two hits, for the beginning and end of *washington*, and two false positives, for splitting where it should not. Hit rate and false positive rate are often combined in a single statistic called F [35]:

$$\mathrm{F} = \frac{2 \times \text{False positive rate} \times \text{Hit rate}}{\text{False positive rate} + \text{Hit rate}}$$

Generally, higher F measures indicate better overall performance.

4.2 Comparison conditions

Several of the following experiments compare the results of Voting Experts with those given by the Sequitur algorithm. Sequitur is an unsupervised, compression-based algorithm that builds a context-free grammar from a sequence of discrete tokens [26]. (For a general introduction to stochastic context-free grammars see [20].) It has successfully identified structure in both text and music. This structure is described by the rules of the induced grammar. Each non-terminal in the grammar covers a subsequence of tokens and so specifies two boundaries. Because non-terminals can expand into sequences containing other non-terminals, we fix the level of non-terminal expansion and stipulate that all non-terminals must be expanded to that level. In our experiments, expanding only the rule associated with the start symbol – what we refer to as level 1 expansion – frequently gives the highest F-measure.

It also is instructive to compare Voting Experts's performance with random segmentations. One approach is to split the sequence randomly subject to the constraint that

Algorithm	F-measure	Recall	Precision	Exact %	Dangling %	Lost %
Voting Experts	.76	.75	.76	.56	.39	.05
Sequitur	.58	.58	.57	.30	.56	.14
All-Locations	.36	1.0	.22	1.0	0.0	0.0
Random-Sample	.21	.22	.21	.05	.34	.61

Table 1: Results of running four different algorithms on George Orwell's *1984*.

there are as many induced words as true words. We call this procedure Random-Sample. Another approach is to place a boundary at every location. We call this procedure All-Locations.

4.3 Voting Experts Applied to Word Boundaries

We removed spaces and punctuation from texts and assessed how well Voting Experts induced word boundaries. To give a sense of its level of performance, here is Voting Experts's segmentation of the first 500 characters of George Orwell's *1984*. The ⋆ symbols are induced boundaries.

Itwas ⋆ a ⋆ bright ⋆ cold ⋆ day ⋆ in ⋆ april ⋆ andthe ⋆ clockswere ⋆ st ⋆ ri ⋆ king ⋆ thi ⋆ rteen ⋆ winston ⋆ smith ⋆ his ⋆ chin ⋆ nuzzl ⋆ edinto ⋆ his ⋆ brea ⋆ st ⋆ in ⋆ aneffort ⋆ to ⋆ escape ⋆ the ⋆ vilewind ⋆ slipped ⋆ quickly ⋆ through ⋆ the ⋆ glass ⋆ door ⋆ sof ⋆ victory ⋆ mansions ⋆ though ⋆ not ⋆ quickly ⋆ en ⋆ ought ⋆ oprevent ⋆ aswirl ⋆ ofgrit ⋆ tydust ⋆ from ⋆ ent ⋆ er ⋆ inga ⋆ long ⋆ with ⋆ himthe ⋆ hall ⋆ ways ⋆ meltof ⋆ boiled ⋆ cabbage ⋆ and ⋆ old ⋆ ragmatsa ⋆ tone ⋆ endof ⋆ it ⋆ acoloured ⋆ poster ⋆ too ⋆ large ⋆ for ⋆ indoor ⋆ dis ⋆ play ⋆ hadbeen ⋆ tack ⋆ ed ⋆ tothe ⋆ wall ⋆ it ⋆ depicted ⋆ simplya ⋆ n ⋆ enormous ⋆ face ⋆ more ⋆ than ⋆ ametre ⋆ widethe ⋆ faceof ⋆ aman ⋆ of ⋆ about ⋆ fortyfive ⋆ witha ⋆ heavy ⋆ black ⋆ moustache ⋆ and ⋆ rugged ⋆ ly ⋆ handsome ⋆ featur

The segmentation is imperfect: Words are run together (*Itwas*, *aneffort*) and broken apart (*st ⋆ ri ⋆ king*). Occasionally, words are split between segments (*to* in *en ⋆ ought ⋆ oprevent*). Still, the algorithm finds 75% of the true word boundaries (recall) and 76% of the boundaries it finds are correct (precision) for an F measure of 0.76. Even errorful segmentation sometimes produces real words or morphemes (though not the ones that appear in the text above). Note that this particular run of Voting Experts involved building an ngram tree of depth 7 and a window of length 6, which means the algorithm correctly finds long words it has never seen, either as ngrams or within its window; words like cabbage, handsome, and moustache.

4.3.1 English

We ran Voting Experts, Sequitur, and both control conditions on the first 50,000 characters of Orwell's *1984*. The detailed results are given in Table 1. These results were obtained with a window length of 7 and a threshold for splitting of three votes. The algorithm induced 11040 boundaries, for a mean induced word length of 4.53. The mean true word length is 4.49. As noted above, the recall and precision are 0.75 and 0.76, respectively. Exact cases, described above, constitute 56% of all cases; that is, 56% of true

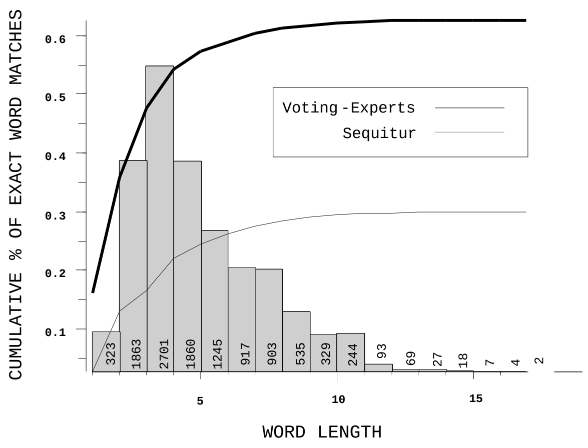

Figure 4: A comparison of cumulative exact match-rate over word length for SEQUITUR and VOTING EXPERTS. The background histogram depicts the distribution of word lengths in the Orwell corpus.

words were bounded at both ends by induced boundaries. Dangling and lost cases constitute 39% and 5% of all cases, respectively. Said differently, only 5% of all true words got lost between induced word boundaries. These tend to be short words, in fact, 59% of the lost words have length 3 or less and 85% have length 5 or less. 89% of the words for which the algorithm found exact boundaries have length 3 or longer. Clearly, VOTING EXPERTS does best with medium-length and long words.

SEQUITUR performs best by expanding its rules only to level 1, that is, by not further expanding any non-terminals off the sentential rule. Further expansions increase the false positive rate. For example, when expanding to level 5, SEQUITUR identifies 78% of the word boundaries correctly, but it induces so many boundaries that its false positive rate is 68%, approaching the 78% false positive rate incurred by splitting at *every* location.

SEQUITUR finds *frequent* patterns, but these do not always correspond to words. Moreover, running with the frequency expert *alone*, VOTING EXPERTS gets recall, precision and F measures of 0.55, 0.75 and 0.64, respectively.

In Figure 4 the cumulative percentage of exact word matches is plotted as a function of word lengths. The distribution of word lengths is shown in the background of the figure. The slope of the cumulative percentage curve is steeper for VOTING EXPERTS than it is for SEQUITUR in the interval corresponding to short and medium-length words, and these words make up most of the corpus. Where accuracy matters most, on the most common words, VOTING EXPERTS is most distinct from SEQUITUR.

Two additional control conditions were run and yielded the expected results. First, VOTING EXPERTS performs marginally less well when it is required to segment text it has not seen. For example, if the first 10,000 characters of Orwell's text are used to build the ngram trie, and then the algorithm is required to segment the next 10,000 characters, there is a very slight decrease in performance. Second, if the "true words" in a text are actually random subsequences of letters, then VOTING EXPERTS performs very poorly.

The effects of the corpus size and the window length are shown in Figure 5. The

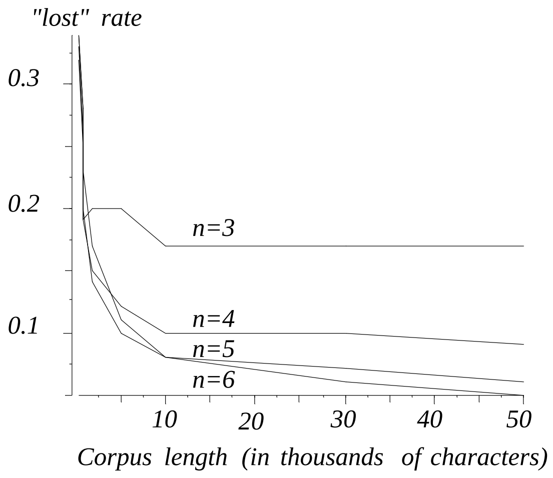

Figure 5: The proportion of "lost" words as a function of corpus length.

proportion of "lost" words (cases 3 and 6, above) is plotted on the vertical axis, and the corpus length is plotted on the horizontal axis. Each curve in the graph corresponds to a window length, k. The proportion of lost words becomes roughly constant for corpora of length 10,000 and higher. The positive effect of increasing window length is probably due to the larger number of "looks" each location gets with larger windows. Given a window of length n, each expert makes n estimates of whether a location is a word boundary. These estimates are not independent, yet each is based on slightly different information, because the ngrams surrounding the location change as the window moves over the location. Apparently, Voting Experts performs better as it makes more estimates of whether a location is a boundary, although the benefit of increasing window length diminishes.

4.3.2 Chinese, German and Roma-ji

As a test of the generality of Voting Experts, we ran it on corpora of Roma-ji, Chinese and German texts. Roma-ji is a transliteration of Japanese into roman characters. The Roma-ji corpus was a set of Anime lyrics comprising 19163 characters. The Chinese text comes from Guo Jim's Mandarin Chinese PH corpus. The PH corpus is taken from stories in newspaper texts and is encoded in in the standard GB-scheme. Franz Kafka's *The Castle* in the original German comprised the final text. For comparison purposes we selected the first 19163 characters of Kafka's text and the same number of characters from *1984* and the PH corpus. As always, we stripped away spaces and punctuation. The window length was 6. The results are given in Table 2.

Clearly the algorithm is not biased to do well on English. In particular, it performs very well on Kafka's text, losing only 4% of the words and identifying 61% exactly. The algorithm performs less well with the Roma-ji text; it identifies fewer boundaries accurately (it places 34% of its boundaries within words) and identifies fewer words exactly. Voting Experts performed worst on the Chinese corpus. Only 42% of the boundaries were identified, although the false positive rate is an extremely low 7%. The explanation for these results has to do with the lengths of words in the corpora. We know that the algorithm

Voting Experts	F-measure	Hit Rate	F.P. Rate	Exact %	Dangling %	Lost %
German	.75	.79	.31	.61	.25	.04
English	.71	.76	.33	.58	.38	.04
Roma-ji	.65	.64	.34	.37	.53	.10
Chinese	.57	.42	.07	.13	.57	.30

Table 2: Results of running Voting Experts on Franz Kafka's *The Castle*, Orwell's *1984*, a subset of the Chinese PH corpus of newspaper stories, and a set of Roma-ji Anime lyrics.

loses disproportionately many short words. Words of length 1 or 2 make up 93% of the Chinese corpus. Words of length 2 make up 39% of the Chinese corpus, 32% of the Roma-ji corpus, 17% of the Orwell corpus, and 10% of the Kafka corpus, so it is not surprising that the algorithm performs worst on the Chinese corpus and best on the Kafka corpus.

If we compensate for the fact that Chinese words are short by decreasing the splitting threshold, we can increase the F-measure of Voting Experts to 77% on the PH corpus. In general, knowing the true mean word length can help. In fact, one can do away with the threshold number of votes and replace it with a much better mechanism: Given the average word length and the corpus size, we know n_w, the number of words in the corpus. Now rank every location by the number of votes it receives select the top-ranked n_w locations, and insert boundaries at these locations. This method increases the accuracy of the algorithm and also increases the fraction of words it finds that are true words.

4.4 Robot Episodes

Whether one works with letters, words, or phrases, text is a sequence of individual symbols. In many applications — robotics, credit card transaction behavior, notational analysis in sports — the data are sequences of vectors of symbols, $\mathbf{x}_t$. For instance, $\mathbf{x}_t$ might be a vector of attributes of a credit card transaction, such as whether the cardholder was present and the vendor code. One can run Voting Experts on such data in two distinct ways: First, one can hash the vector to a single symbol and run the algorithm as usual on the sequence of these symbols; or, one can run the algorithm on the sequence of each element of the data vector and somehow combine the votes for splitting at location t in each sequence.

We tested both methods with data generated by a mobile robot, a Pioneer 2 equipped with sonar and a pan-tilt-zoom camera running a subsumption architecture. In each time series the robot wandered around a large playpen for 20-30 minutes looking for interesting objects, which it would orbit for a few minutes before moving on. At one level of abstraction, the robot engaged in four types of behaviors: *wandering, avoiding, orbiting* and *approaching.* Each behavior was implemented by sequences of actions initiated by *controllers* such as *move-forward* and *center-camera-on-object.* The challenge for Voting Experts was to find the boundaries of the four behaviors given only information about which controllers were on or off. This problem is like finding words given letters, except there are just eight controllers. One way to represent the states of eight controllers is with a binary vector of length eight, the other is to transform the bits in the vector to a single number between 1 and 256. In fact, only 65 of the possible 256 combinations of controllers manifested themselves during

	F	Recall	Precision	Exact %	Dangling %	Lost %
Summer univariate	0 .778	0.774	0.781	0.323	0.479	0 .196
Summer multivariate	0.683	0.558	0.880	0.040	0.378	0.574
Autumn univariate	0.183	0.651	0.115	0.030	0.580	0.370
Autumn multivariate	0.197	0.500	0.123	0.039	0.438	0.514

Table 3: Results of running Voting-Experts on the Summer and Autumn data sets, hashing multivariate states to univariate symbols as well as running Voting-Experts on the individual streams.

the experiment. Thus we may run Voting Experts on either a sequence of individual symbols — the labels 1, 2, ..., 65 — or separately on eight binary sequences. In either case, the task is to find the places in the robot's history where it shifts between its four behaviors.

The Summer dataset is 17798 time steps long and has 2237 changes in behavior. The mean duration of a behavior is 7.95 time steps. We also ran the robot under different controllers to get a second dataset with very different statistics, specifically, much longer-lasting behaviors. The Autumn dataset had 8923 ticks and 105 changes of behavior, with mean episode length 84.97.

The definition of hit rate and false positive rate were changed slightly for these experiments. Because the controller data can be noisy at the episode boundaries, we classify as *hits* induced boundaries within one time step in either direction of when a behavior actually changed. A false-positive is recorded only when a boundary is induced and no real boundary exists within a window of one time step on either side of the induced boundary.

Results are shown in Table 3. Clearly, performance on the Summer data is comparable to that for words in text, but performance on the Autumn data is very much worse. This is because very long episodes are unique, so most locations in very long episodes have zero boundary entropy and frequency equal to one. Also, if the window size is very much smaller than the episode size then there will be a strong bias to cut the sequence inappropriately. It makes little difference whether we run the algorithm on a single sequence of symbols in the range 1..65 or on eight individual binary sequences (one for each controller) and cut the sequences at the locations that have the most votes summed across the sequences.

5 Related work

The psychological literature on chunks is largely descriptive and does not characterize intrinsic properties of chunks such as within- and between-chunk entropy (though see [31]). One can get lost quickly in the literature on chunking and memory; good places to begin are de Groot's classic experiments [9] on chess masters' memory for chess positions (see also Chase and Simon [4] and Gobet [13]). Chunking is the principal learning mechanism in the SOAR architecture [22].

Segmentation is related to chunking and has received much attention over the years, particularly in linguistics. Even before Miller's classic article, Zelig Harris noted that the unpredictability of phonemes was higher at word boundaries than elsewhere [16], and, before that, Shannon recognized the same phenomenon in his experiments on the entropy of English [30]. Hafer and Weiss [15] developed an algorithm similar to VOTING EXPERTS. They used a trie to represent ngrams (as we do) and they split sequences based on boundary entropy, but their method did not work very well, in part because they did not take frequencies into account and they did not standardize frequencies, an important step explained in Section 3.1. Some researcher try similar approaches to finding morphemes rather than words (e.g., [8]). In fact, many of VOTING EXPERTS's errors involve cutting words at morphological boundaries. Plurals are especially difficult, so are prefixes and suffixes such as "re" and "ing". A large part of the problem of morphological analysis is figuring out that "re" is a prefix in "rerun" but not "read" (e.g., [14]). Another problem is the lack of agreement on correct morphological decomposition. Some corpora we have seen are at best debatable.

The influential work of Saffran, Aslin and Newport [28] showed that eight-month-old infants could segment synthetic speech lacking prosodic cues and generated from an artificial grammar with a novel vocabulary. Remarkably, they could do this after only two minutes exposure to the speech. Saffran and her colleagues constructed their grammar so that the conditional probabilities of adjacent symbols were low when the symbols were on either side of a word boundary and higher within words. Thus, the words in their grammar had low within-word entropy and high between-words entropy, and infants picked up on this signature almost immediately. (It should be noted that the segmentation task was not very challenging: VOTING EXPERTS performed it perfectly.) Hauser, Newport and Aslin [17] report much the same results with cotton-top tamarins.

One notable conclusion from the cognitive literature is that if chunking works in infants and monkeys, it cannot require a great deal of knowledge and it probably gets to work very early in life.

There are numerous applications of chunking and segmentation, many in computational linguistics and genomics. Some of these are presented in detail, later. The problem of finding chunks in sequences of medical insurance claims data is addressed in [37]; robot gripping episodes are learned by [38]; and sequences of agent interactions are learned by [18]. We demonstrated VOTING EXPERTS's performance on a robotic episode problem in Section 4.4.

Computer scientists have developed dozens or hundreds of algorithms for segmentation and related tasks, such as compression. Some are similar to VOTING EXPERTS, or different in instructive ways. The best segmentation algorithms are trained with positive instances of segmented text (e.g., [24, 12, 36]) or use a supervised form of compression (e.g., [33, 34]). In contrast VOTING EXPERTS is unsupervised, meaning it gets text from which all indicators of boundaries have been removed. Ando and Lee describe an unsupervised algorithm for segmenting Kanji Japanese script. Their algorithm tries to maximize the entropy between words and minimize the entropy within (although Ando and Lee do not present it this way) by requiring that two adjacent ngrams have higher frequency than one ngram "in the middle." For example, in a sequence *abcd*, if the frequencies of *ab* and *cd* are high relative to the frequency of *bc*, this suggests breaking the sequence between *b* and *c*. We applied this algorithm to the same texts as VOTING EXPERTS and it performed less well than the frequency and boundary entropy methods. We added Ando and Lee's heuristic to VOTING EXPERTS as a third method and did not find it contributed to consistent or non-negligible improvements in performance. Ando and Lee's heuristic appears to be getting at the same signature of chunks as frequency and boundary entropy are, though less effectively.

Parsing or segmentation based on mutual information (e.g., [23, 21]) is similar to segmenting with boundary entropy. The idea is simply to measure the mutual information between the left and right halves of successive bigrams and cut the sequence where the mutual information peaks. This method attends to the between-chunk entropy aspect of chunks, not the within-chunk entropy. It is like running VOTING EXPERTS with only the boundary entropy method. Performance is good, but not as good as with both.

Compression seems a natural way to find chunks (e.g., [19, 33, 34, 26]. We compared VOTING EXPERTS with the SEQUITUR compression-based segmentation algorithm in Section 4.3. Both are unsupervised algorithms and, not surprisingly, neither performs as well as the supervised compression-based method of Teahan [33, 34]). VOTING EXPERTS did better than SEQUITUR in our tests, probably because, like other compression algorithms, SEQUITUR attends exclusively to frequency. As noted earlier, frequency is equivalent to within-chunk entropy. It is worthwhile to attend also to between-chunk entropy.

Several authors try to segment text by a kind of maximum-likelihood parsing. They first learn an ngram language model, then segment text so as to maximize the probability of the observed text given the model. Examples of this approach include [2, 10, 39]. These methods all search over all possible segmentations of subsequences of the text, looking for the segmentation that maximizes probability according to a language model that gets built as the algorithm runs. Because Brent's MBDP-1 is a very clear representative of this approach, we will consider it in some detail here.

Brent's MBDP-1 algorithm [2] simultaneously learns words and segments text, using the segmentation to identify new words and the probabilities of previously learned words to aid segmentation. It is exposed to sentences one at a time, segments each, adds new words to its lexicon, and iterates. This bootstrapping aspect distinguishes MBDP-1 from VOTING EXPERTS, which works with a batch corpus, only. Brent argues that for a segmentation algorithm to be a plausible model of word-learning, it must work incrementally. We agree, so we ran a very crude approximation to Brent's incremental approach with VOTING EXPERTS. Following Brent, we used the English version of the Canadian Hansard corpus as a source. Brent presents learning curves for recall and precision as functions of the amount of text presented to MBDP-1 [3]. He reports performance at $2^2, 2^4, 2^6, \ldots, 2^{22}$ words. We ran VOTING EXPERTS with corpora of 6×2^i letters, for $i = 2, 4, 6, \ldots, 22$, as the Hansards have 5.85 letters per word, on average. Figure 6 shows that VOTING EXPERTS has a flatter curve than MBDP-1 and actually performs better with small corpora, but for corpora of 2^{14} words (nearly 100,000 characters) and higher, MBDP-1 marches on to roughly perfect recall and precision while VOTING EXPERTS remains at roughly 90% recall and 65% precision.

Why the dfference? One possibility is that MBDP-1 makes "very good use of sentence boundaries and punctuation" [2], neither of which is available to VOTING EXPERTS. With every sentence, MBDP-1 gets one guaranteed instance of the beginning of a word and one guaranteed instance of the end of a word (the first and last words in the sentence). For corpora of 2^{18} words, assuming eight words per sentence, this amounts to $32,000$ known beginnings and ends of words. However, Brent suggests that the value of this information is most evident early in the learning process, and we agree that that is where it is likely to have the best effect on word probabilities, so it probably is not the explanation of the

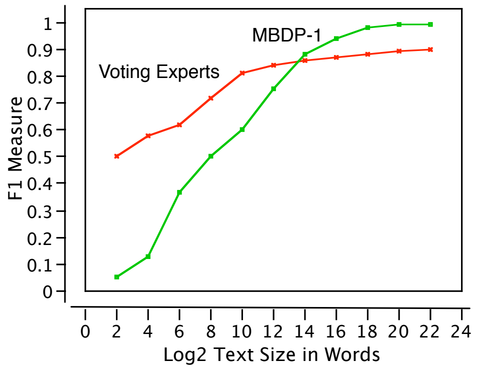

Figure 6: F1 measures for MBDP-1 and Voting Experts

difference between the algorithms at large corpus sizes.

Note that MBDP-1 truly is a bootstrapping algorithm, in that previous segmentations refine word probabilities, which help later segmentations; whereas Voting Experts merely segments the entire corpus and does not form any word models. The question is, is bootstrapping a superior approach? It is intuitively appealing to use the words one has learned by segmentation to help learn new words, but does account for near-perfect performance at large corpus sizes, where Voting Experts stalls? Recently, we have tried saving the words discovered by Voting Experts and using them to improve later segmentation, but we have not been able to significantly improve the performance of our algorithm. For us, it remains an open question whether bootstrapping is superior to Voting Experts simple method of looking for the signature of low within-chunk entropy and high between-chunk entropy.

6 Discussion

The Voting Experts algorithm depends on two indicators of chunk boundaries: the entropy of the element that follows a potential boundary, and the sum of the standardized frequencies of ngrams to the left and right of a potential boundary. These experts do not always agree. In fact, as Voting Experts passes its window over the Orwell text, the experts vote for the same location in the window only about 45% of the time. Moreover, Voting Experts performs quite well with either expert alone, generally coming within a few percentage points of the F measures achieved with both experts. We have added numerous other experts to the mix; none changes our results by more than a few percentage points. We have tried different ways to adjust or smooth the votes cast by the experts and different methods for deciding where to split the sequence given the pattern of votes; Cheng and Mitzenmacher propose a different scheme for combining votes [5]. Again, none of these variants have large effects on performance.

These explorations give rise to two conjectures: First, all the experts we implemented are variants of one signature: entropy within chunks is low, entropy between chunks is high. Second, the performance of Voting Experts and all its variants is limited by the structure of the sequences it segments, and fiddling with the algorithm will not improve its performance by much. In particular, orthographies of natural languages are not prefix codes, and that many chunking errors are due to one word being embedded in another (e.g., the sequence *...nothere...* might be part of *...no there...* or *...not here...* or *...not her elephant...* or *no the red one...* and so on). Said differently, Voting Experts might be performing as well as possible, given the structure of natural language text. A simple experiment tends to bear this out: Many subsequences are repeated in text (e.g., "there" appears often) and, without contextual information, Voting Experts necessarily cuts each replication of such a subsequence at the same place. Usually, the algorithm's choice is correct, but it is rarely correct for *all* the instances of a given subsequence. This means Voting Experts simply cannot do better without increasing the length of subsequences to incorporate more context.

7 Conclusion

Voting Experts attends to a statistical signature of chunks — low entropy within, high entropy between — to achieve relatively high recall and precision on segmentation tasks. The algorithm approaches these tasks in a way unlike anything that humans or other animal would naturally do. It works with a large batch of meaningless symbols. It has no purpose other than to compute between– and within–chunk entropy. It does not hypothesize chunks and test them in a performance task, it is entirely unsupervised and data-driven. That it performs so well speaks to the power of the signature of chunks. Human chunking, however, is an incremental process in which previously-learned chunks help the learner recognize and learn new chunks (i.e., it is a bootstrapping process, as Brent [2] asserts). Most importantly, chunks are formed for a reason. Why does the mind form chunks? How are chunks different from the elements from which they are formed? How are words different from morphemes, and how are phrases different from words? One argument for chunking is *cognitive economy* — compression, essentially — the idea that by putting elements together one could think in terms of relatively few chunks instead of relatively many elements. (Miller called this recoding: "[P]robably the simplest [way] is to group the input events, apply a new name to the group, and then remember the new name rather than the original input events." [25]) While cognitive economy might be the reason for chunks, it cannot be the reason for *particular systems of chunks.* Said differently, a corpus of elements can be chunked in a variety of ways, all of which afford cognitive economy, but only some of which are preferred.

Our current research on chunking involves asking why some chunks are preferred. Our idea is that chunks are preferred because they provide qualitatively different functionality than one gets from elements of chunks. The mental operations performed on words are qualitatively different from those performed on phrases. One can compare the meanings of words (e.g., "happy" and "sad"), whereas one can reason about the truth and entailments of phrases (e.g., "George is happy"). The different levels of language, particularly the word and phrase levels, differ in the sorts of mental operations or functions they enable. The *developmental chunking principle* is that chunks are formed at one level to improve performance on a task at a higher level. This principal is the focus of our ongoing research.

Acknowledgments

We are grateful to Ms. Sara Nishi for collecting the corpus of Anime lyrics, and Aram Galstyan and Wesley Kerr for helpful discussions. This research is supported by DARPA under contract numbers DARPA/USASMDCDASG60-99-C-0074 and DARPA/AFRLF30602-01-2-0580. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of DARPA or the U.S. Government.

References

[1] R. K. Ando and L. Lee. Mostly-unsupervised statistical segmentation of japanese: Application to kanji. In *Proceedings of North American Association for Computational Linguistics (NAACL)*, pages 241–248, 2000.

[2] M. R. Brent. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 45(1–3):71–105, 1999.

[3] M. R. Brent and X. Tao Chinese text segmentation with MBDP-1: making the most of training corpora Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Toulouse, France. pp. 90 - 97 . 2001

[4] Chase, W. G., and Simon, H. A. The mind's eye in chess. In W. G. Chase (Ed.), Visual information process- ing (pp. 215-281). New York: Academic Press. (1973)

[5] Jimming Cheng, Michael Mitzenmacher The Markov Expert for Finding Episodes in Time Series Unpublished technical report

[6] P. Cohen and N. Adams. An algorithm for segmenting categorical time series into meaningful episodes. In *Proceedings of Fourth Symposium on Intelligent Data Analysis*, pages 198–207. Springer, 2001.

[7] P. R. Cohen, B. Heeringa, and N. Adams. An unsupervised algorithm for segmenting categorical time series into episodes. In *Working Notes of the 2002 ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining (to appear)*, 2002.

[8] Mathias Creutz Unsupervised segmentation of words using prior distributions of morph length and frequency. In Proceedings of ACL-03, the 41st Annual Meeting of the Association of Computational Linguistics, pages 280-287, Sapporo, Japan, 7-12 July.

[9] de Groot, A. D. Thought and choice in chess (1st ed.). The Hague: Mouton Publishers. 1965.

[10] de Marcken, C. Unsupervised acquisition of a lexicon from continuous speech. Technical Report AI Memo No. 1558, Massachusetts Institute of Technology, Cambridge, MA. 1995.

[11] W. Fitch and M. Hauser. Computational constraints on syntactic processing in a nonhuman primate. *Science*, 303(5656):377–380, 2004.

[12] M. N. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In *The VLDB Journal*, pages 223–234, 1999.

[13] Gobet, F. and Simon, H. A. (1996). The roles of recognition processes and look-ahead search in time-constrained ex- pert problem solving: Evidence from grandmaster level chess. Psychological Science, 7, 52-55.

[14] John Goldsmith Linguistica: An automatic morphological analyzer To appear in John Boyle, Jung-Hyuck Lee, and Arika Okrent CLS 36 [Papers from the 36th Meeting of the Chicago Linguistics Society]Volume1: The Main Session. 2000.

[15] M. A. Hafer and S. F. Weiss. Word segmentation by letter successsor varieties. Information Storage and Retrieval, 10(371385). 1974.

[16] Zelig Harris Distributional Structure, Word 10 (1954): 156.

[17] M. D. Hauser, E. L. Newport, and R. N. Aslin Segmentation of the speech stream in nonhuman primates: Statistical learning in cotton top tamarins Cognition. 78. 2001

[18] Yoav Horman and Gal A. Kaminka Improving Sequence Recognition for Learning the Behavior of Agents In *Proceedings of the Association AAMAS 2004.*

[19] J. L. Hutchens and M. D. Alder. Finding structure via compression. In D. M. W. Powers, editor, *Proceedings of the Joint Conference on New Methods in Language Processing and Computational Natural Language Learning: NeMLaP3/CoNLL98*, pages 79–82. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[20] Frederick Jelinek, John D. Lafferty, and Robert L. Mercer. Basic methods of probabilistic context free grammars. In Pietro Laface and Renato De Mori, editors, Speech Recognition and Understanding. Recent Advances, Trends, and Applications, volume F75 of NATO Advanced Sciences Institutes Series, pages 345-360. Springer Ver- lag, Berlin. Proceedings of the NATO Advanced Study Institute, Cetraro, Italy, July 1990.

[21] A. Kempe. Experiments in unsupervised entropy-based corpus segmentation. In *Proc. Workshop on Computational Natural Language Learning (CoNLL)*, pages 7–13, Bergen, Norway, June 1999. SIGNLL/ACL.

[22] Laird, J. E., Rosenbloom, P. S. and Newell, A. Chunking in Soar: The anatomy of a general learning mechanism. Machine Learning 1(1):11-46, 1986.

[23] D. Magerman and M. Marcus. Parsing a natural language using mutual information statistics. In *Proceedings, Eighth National Conference on Artificial Intelligence (AAAI 90)*, pages 984–989, 1990.

[24] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

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

[26] C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7(1):67–82, 1997.

[27] J. T. Oates. *Grounding Knowledge in Sensors: Unsupervised Learning for Language and Planning.* PhD thesis, Department of Computer Science, University of Massachusetts, 2001.

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

[29] P. K. Sahoo, S. Soltani, A. K.C. Wong, and Y. C. Chen A survey of thresholding techniques Computer Vision, Graphics, and Image Processing archive Volume 41 , Issue 2 Pages: 233 - 260. 1988

[30] Claude Shannon Prediction and the Entropy of Printed English Bell System Technical Journal, 1951

[31] H. A. Simon How Big is a Chunk? Science, Volume 183, Issue 4124, pp. 482-488, 1974

[32] Andreas Stolcke. *Bayesian Learning of Probabilistic Language Models.* PhD thesis, Division of Computer Science, University of California, Berkeley, 1994.

[33] W. J. Teahan, Y. Wen, R. J. McNab, and I. H. Witten. A compression-based algorithm for chinese word segmentation. *Computational Linguistics*, 26(3):375–393, 2000.

[34] W. J. Teahan. Text classification and segmentation using minimum crossentropy. In International Conference on Content-Based Multimedia Information Access (RIAO), 2000.

[35] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[36] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Knowledge Discovery and Data Mining*, pages 359–363, 1998.

[37] Graham Williams, Chris Kelman, Rohan Baxter, Lifang Gu,Simon Hawkins, Hongxing He, Chris Rainsford, and Deanne Vickers. Temporal Event Mining of Linked Medical Claims Data". PaKDD 03.

[38] Richard Alan Peters II, Christina L Campbell, William J Bluethmann and Eric Huber Robonaut Task Learning through Teleoperation In *Proceedings of the IEEE International Conference on Robotics and Automation*

[39] Anand Venkataraman A Statistical Model for Word Discovery in Transcribed Speech Computational Linguistics Vol. 27, No. 3, Pages 351-372 2001