Memory constraints affect statistical learning; statistical learning affects memory constraints.

Joshua R. de Leeuw (jodeleeu@indiana.edu)
Department of Psychological and Brain Sciences & Program in Cognitive Science
Bloomington, IN 47405 USA

Robert L. Goldstone (rgoldsto@indiana.edu)
Department of Psychological and Brain Sciences & Program in Cognitive Science
Bloomington, IN 47405 USA

Abstract

We present evidence that successful chunk formation during a statistical learning task depends on how well the perceiver is able to parse the information that is presented between successive presentations of the to-be-learned chunk. First, we show that learners acquire a chunk better when the surrounding information is also chunk-able in a visual statistical learning task. We tested three process models of chunk formation, TRACX, PARSER, and MDLChunker, on our two different experimental conditions, and found that only PARSER and MDLChunker matched the observed result. These two models share the common principle of a memory capacity that is expanded as a result of learning. Though implemented in very different ways, both models effectively remember more individual items (the atomic components of a sequence) as additional chunks are formed. The ability to remember more information directly impacts learning in the models, suggesting that there is a positive-feedback loop in chunk learning.

Keywords: statistical learning; chunking; memory

Introduction

The formation of chunks is hypothesized to be a crucial aspect of cognition, perception, and learning (Gobet et al., 2001). Chunks are a means of creating compressed encodings for frequently co-occurring inputs. The concept of chunking has been used to explain a wide range of psychological phenomena, including the advantages that expert chess players have in remembering the position of chess pieces on a board (Chase & Simon, 1973; Gobet & Simon, 1998), differences in the speed of retrieving successive letters of the alphabet (Klahr, Chase, & Lovelace, 1983), and the ability to remember more words when the words are part of familiar phrases (Simon, 1974). A core aspect of chunking is that it increases the number of items that can be stored in memory: It is possible to remember more individual letters if they are chunked into words, and more words if they are chunked into sentences.

Statistical learning paradigms are well suited for investigating the conditions under which chunks are learned (Perruchet & Pacton, 2006). In a typical statistical learning task, a novel information stream containing latent structure is presented to a subject for a moderate length of time, and the subject is tested on how well they are able to learn the structure that generated the stream. Often this structure is explicitly in the form of chunks (e.g. Fiser & Aslin, 2001, 2002).

A key part of statistical learning research is identifying the conditions under which chunking occurs. The foundational work focused on learning based on transitional probabilities (Aslin, Saffran, & Newport, 1998; Saffran, Aslin, & Newport, 1996), and much subsequent research has explored different constraints and biases that affect learning. A key theme from this research is that previous learning experience alters how new information is processed. Learners form expectations about the kind of structure that is present in an information stream from previous exposure to other streams (Lew-Williams & Saffran, 2012). This can cause them to fail to learn structures that are in conflict with their expectations (Gebhart, Aslin, & Newport, 2009). Prior learning can also improve subsequent learning. For example, acquiring non-adjacent dependencies is easier after first learning the adjacent dependencies (Lany & Gómez, 2008).

Memory constraints are an important factor in determining the success of learning new chunks. Frank and Gibson (2011) showed that statistical rule learning is improved in a variety of experimental paradigms when memory constraints are alleviated by presenting examples concurrently instead of sequentially. They hypothesize that this is because learners need to be able to remember enough items in order to extract the statistical regularities. However, it is unknown what functional role the memory constraints might play.

Models of statistical learning vary on whether they include memory constraints and how such constraints are implemented. Models with memory constraints, either in terms of a limit on the number of input items that can be remembered or a limit on the number of internal states that the model can track, tend to fit human performance on word segmentation tasks better than models without such constraints (Frank, Goldwater, Griffiths, & Tenenbaum, 2010). However, previous models have not explored how the learning process and the memory constraints might interact. Since statistical learning is hypothesized to involve chunk formation, and chunks are more efficient memory structures for encoding information, learning may have a cyclical effect: learning to chunk may reduce the memory constraints of encoding a sequence, allowing people to

remember more items and more easily extract the regularities. We tested this hypothesis in a simple experiment.

Method

We replicated and extended a classic result from temporal visual statistical learning (Fiser & Aslin, 2002). In the original experiment, subjects were exposed to a sequence of shapes, presented one at a time, with no overt task. Unbeknownst to the subjects, the sequence was formed by grouping the shapes into sets of three items (triples) and presenting the triples in a random order. We replicated the original experiment as a control condition, and also tested subjects' ability to learn an individual triple when the other triples were scrambled. In both conditions, the target triple appears equally often and with equal frequency throughout the sequence. If learning chunks makes it easier to learn other chunks, then learners should show improved learning for the target triple in the condition with more triples.

Participants

41 people participated in the study via Mechanical Turk. Subjects were paid $1.25 for participation. Subjects were randomly assigned into either the *four-triples* (N = 21) or *one-triple* (N = 20) condition.

Procedure

Subjects completed the experiment in a web browser of their choice. The experiment was developed using the jsPsych software library (de Leeuw, 2015).

The experiment consisted of an exposure phase and a test phase. During the exposure phase, subjects viewed a sequence of 300 images with the instructions to simply observe the shapes because they would be asked questions about what they saw. The sequence consisted of 12 unique shapes, modeled after the shapes depicted in (Fiser & Aslin, 2002). The sequence was shown as an animation with shapes oscillating horizontally, moving behind an occluding rectangle in the center of the screen (see Fiser & Aslin, 2002 for a visual depiction). It took one second from the point that a part of the shape appeared to the point that the shape was completely occluded again. The entire sequence lasted five minutes.

In the four-triples condition, the shapes were grouped into four triples, with each shape belonging to one triple (figure 1). The assignment of particular shapes to triples was randomized for each subject. The sequence was created by randomly ordering the triples, with the constraints that: (1) a triple could not occur twice in a row, (2) a triple could not occur more than twice before every other triple occurred at least once, and (3) all triples occurred exactly 25 times.

In the one-triple condition, the sequence was created in a similar way, except that the order of three of the triples was randomized for each presentation of the triple. Thus, if one of the randomized triples was `ABC`, it would randomly appear as `ABC`, `ACB`, `BAC`, `BCA`, `CAB`, and `CBA`. One of the four triples was always presented as a consistent triple, maintaining its original order. In addition, three impossible triples were created for testing purposes. Impossible triples contained one shape from each of the three randomized triples. When the sequence was constructed, shapes that occurred in the same impossible triple could not occur sequentially. This constraint allowed for a comparable test in both conditions: a triple that was seen could be paired with a triple that was never observed.

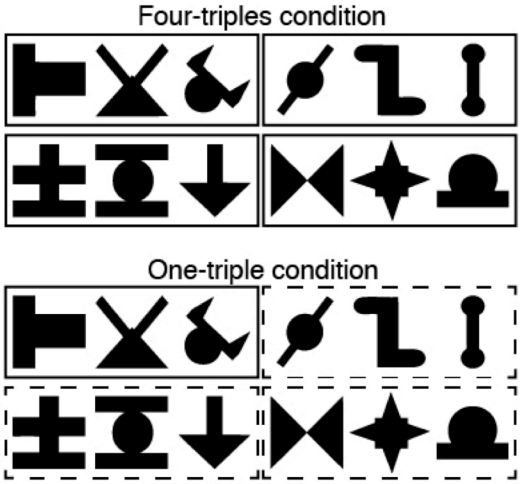

Figure 1: Shape stimuli used in experiment 1. In the four-triples condition, stimuli were grouped into four triples (illustrated with solid boxes). In the one-triple condition, one triple appeared with the shapes in the same order throughout the sequence (solid box), and the rest of the stimuli appeared in groups of three but with a random order of the shapes inside the box during each appearance (dashed boxes)

In the test phase, subjects were sequentially presented with two three-item sequences and asked to report which triple occurred more often during the exposure phase. Each three item sequence was presented in the same manner as the sequence during the exposure phase. There was a 1 second gap between the two test sequences. Subjects were required to choose one of the sequences, even if they were unsure. There were 32 test pairs. In the four-triples condition, four impossible triples were created, where the probability of each item in the triple appearing adjacent to the other items during the exposure phase was 0. Each triple was tested against each impossible triple twice, once with the triple first and once with the triple second. In the one-triple condition, we also created four impossible triples, as well as three low-probability triples. The impossible triples never occurred in the sequence, and the low-probability triples occurred rarely. We did not use any data from the test pairs that contained low-probability triples; they were merely created to make the testing phase the same length in both conditions, and to ensure that the frequency of individual shapes was identical in the testing phase. There

were 24 trials containing low-probability triples, and 8 containing the single triple compared with one of the four impossible triples.

Results

Subjects in the four-triples condition had an overall accuracy of 73.4% at identifying the triple they had seen before in the forced-choice tests, while subjects in the one-triple condition were only 58.8% accurate (Figure 2). Thus, subjects in the four-triple condition were 14.8% more accurate at identifying the target triple, on average.

We used a Bayesian data analysis model to estimate the difference in probability of a correct response between the two conditions. There are numerous reasons to favor Bayesian data analysis over conventional null-hypothesis significance testing (Kruschke, 2011), but a significant advantage in this particular application is the ability to naturally account for the different number of critical trials in each condition (32 for the four-triples condition and 8 for the one-triple condition). Each subject's responses were treated as being generated from a binomial distribution with probability p and number of samples N. For subjects in the four-triples condition, $N=32$, and for subjects in the one-triple condition, $N=8$. We estimated p as the sum of two random variables: $p_{baseline}$ and $p_{difference}$. The baseline component estimated the overall mean probability of a correct response across conditions, and the difference component estimated the magnitude of the difference between conditions. The prior on $p_{baseline}$ was a beta distribution with both shape parameters equal to 1, and the prior on $p_{difference}$ was a normal distribution with the mean equal to 0 and the standard deviation equal to 1. These parameters represent vague priors that are appropriate to the scale of the data. We used MCMC sampling with the runjags R package to find the posterior distribution. The 95% highest-density interval (HDI)[1] for $p_{difference}$ was 6.39% to 22.9%, with a mode of 13.4%. Thus, the model finds strong evidence that the four-triple group did indeed learn the triples better than the one-triple group[2].

Modeling

The experiment found evidence that chunk learning is influenced by more than just the repeated presentation of a consistent set of items. The target triple was learned significantly better when the surrounding information was also generated from a triple-based structure. We tested three well-established process models of statistical learning to see if they predicted the difference in learning that we observed.

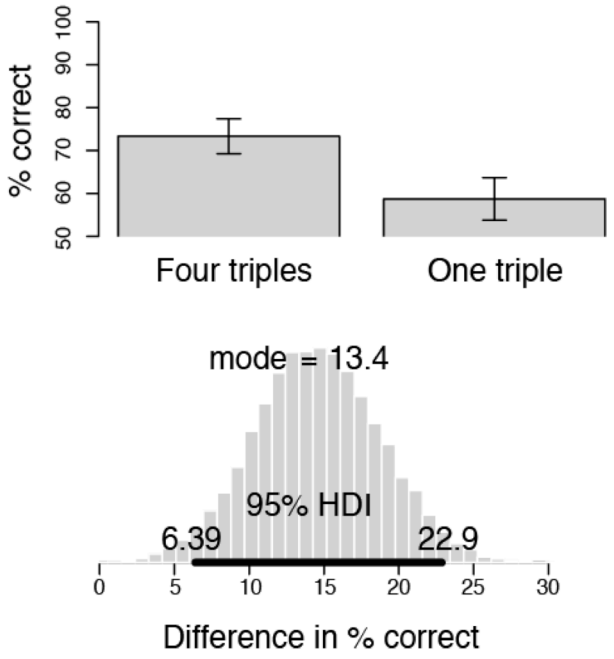

Figure 2: Experiment results. Top: Mean accuracy for the two conditions in experiment 1. Error bars show one standard error of the mean. The y-axis begins at chance performance (50%). Bottom: Posterior distribution of the estimated difference in the probability of a correct answer between conditions. Positive values indicate samples from the posterior in which the four-triples group was more accurate than the one-triple group. The 95% HDI is shown in black, with the limits labeled.

Model descriptions

We tested three models: PARSER, MDLChunker, and TRACX. We chose these models because they are process models that represent different approaches to sequence segmentation and chunk learning, and they all had publically available implementations that we could use. Importantly, the three models all deal with memory constraints in different ways. Here we briefly summarize each model to provide an intuition for how they work. Due to space constraints, please refer to the original source material listed in the heading for a more detailed explanation of each of the models.

PARSER (Perruchet & Vinter, 1998). PARSER constructs an internal lexicon through an online chunk formation process. Candidate chunks are created through a random process as the model processes the input: PARSER selects a percept length of 1, 2, or 3 units (with the default parameter set). This percept becomes a candidate chunk. Frequently seen chunks are reinforced, while candidate chunks that are encountered rarely are forgotten. When the strength of an individual chunk rises above a threshold, then incoming information is shaped by the presence of the

[1] The range of parameter values containing 95% of the posterior where each value inside the HDI is more probable than those outside it. The HDI represents the most likely parameter values for the model given the data.

[2] A t-test of the difference in means also reached the same conclusion of a significant difference in accuracy, $t(39)=2.2855$, $p = 0.028$.

chunk. For example, if the incoming sequence is ABCD and PARSER selects a percept length of 2 and has no chunks, then the model will form a candidate chunk of AB. But if PARSER already has the chunks AB and CD and selects a percept length of 2, then the input sequence ABCD will be processed as AB/CD. This will result in both the AB and CD chunks being reinforced, as well as the formation of a candidate ABCD chunk.

MDLChunker (Robinet, Lemaire, & Gordon, 2011). MDLChunker also creates an explicit internal lexicon, but it uses the minimum description length principle (Rissanen, 1978) to guide the formation of new chunks. As MDLChunker processes a sequence, it checks to see if recoding the sequence using chunks would decrease the number of bits required to encode the sequence. Importantly, adding chunks increases the number of bits required to store the lexicon, and MDLChunker will only add a new chunk if the cost of adding the chunk to the lexicon is outweighed by the overall reduction in coding complexity of the sequence. We used the memory-constrained version of MDLChunker (see section 7.3 of Robinet et al., 2011). Without memory constraints, MDLChunker checks the cost of adding a new chunk against *all* of the input that it has previously seen. The memory constraint imposes a limit, expressed in bits, for how much of the previous input can be retained by the model (and thus used in the calculation for adding a new chunk). Importantly, the memory cost is calculated based on the lexicon. Thus as the model gets more efficient at encoding the input, the absolute number of items in memory will grow.

TRACX (French, Addyman, & Mareschal, 2011). TRACX is a connectionist model of chunk learning. The core of TRACX is an auto-associative network. The input layer represents two adjacent items (called the left- and right-hand items, with the left-hand item occurring temporally before the right-hand item) from the sequence, the hidden layer forms a compressed representation of the input, and the output layer recreates the input. Back-propagation is used to adjust the weights so that the output better matches the input. The key innovation is that the network will use the hidden layer as the left-hand item in the next input when the error in reconstruction is low. Low reconstruction error occurs when the input is very familiar to the network, and thus is a candidate chunk. The distributed pattern of activity on the hidden layer is a representation of the chunk. Initially, TRACX will learn only two-item chunks, but as these chunks are learned and subsequently become part of the input, then longer chunks can also be learned.

Method

Model implementations. We used publicly available implementations of each of the three models. For PARSER and MDLChunker, we used the U-LEARN software from http://perruchet.jimdo.com/u-learn/. For TRACX, we used a JavaScript version of the model from https://github.com/YourBrain/TRACX-Web. We made no modifications to the model code.

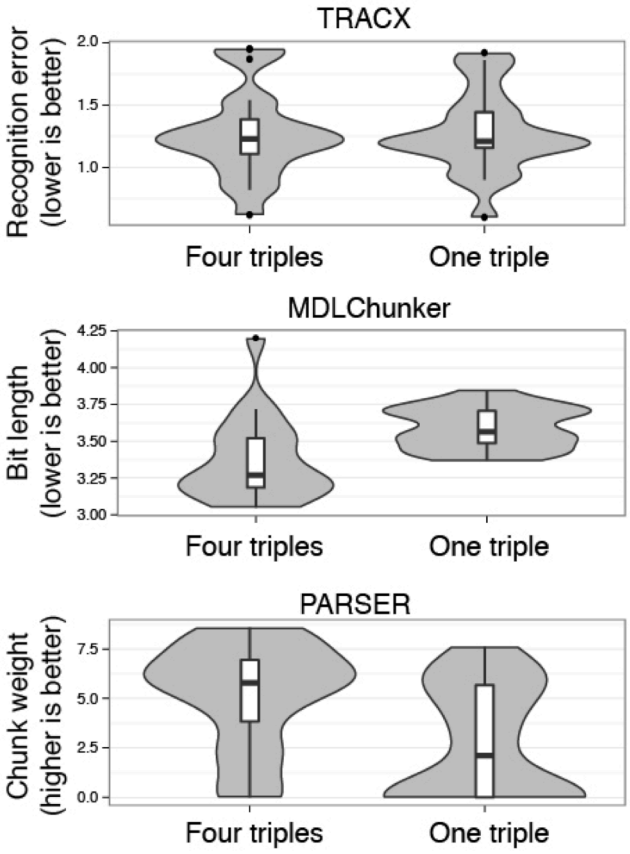

Figure 3: Model results. Each of the three models has a different way of indicating how well the target chunk was learned, indicated on the y-axis. The distributions of the measurement values are shown in grey. PARSER produced a bimodal distribution in the one-triple condition, showing that the target triple was learned only some of the time. The box-and-whisker overlay is provided to show a representation of the central tendency. The dark line is the median, the boxes represent the range of values between the 25th and 75th percentile of the distribution, and the whiskers show the range of data that is within the inter-quartile range (height of the box) times 1.5. Values outside this range are plotted as individual dots.

Procedure. We converted the sequences seen by participants in the experiment into strings of letters, with each shape being represented by a unique letter. The strings were 300 characters long. We used the exact same sequences seen by participants in the experiment. Each model was tested with 20 different four-triple sequences and 20 different one-triple sequences. We used the default parameters for all models.

Each of the three models generates a different kind of output. PARSER and MDLChunker both construct lexicons containing explicit chunks. PARSER assigns a weight to each chunk, with higher scores being chunks that have greater weight. MDLChunker reports the number of bits needed to encode each chunk; smaller bit lengths represent chunks that are more strongly encoded. TRACX produces a network recognition error score for any given input chunk, but because the chunks are represented as distributed patterns there is no list of known chunks produced by the model. Instead, the model is queried with a particular chunk to see what the error rate is. Since our main interest was seeing if any of the models could fit the qualitative pattern and this only requires within-model comparisons, we did not attempt to equate these different output values between models.

It was unclear how to link the various model's outputs to performance on the forced-choice test. TRACX provided a relatively straightforward option, since the recognition error for any particular chunk can be tested. However, both PARSER and MDLChunker will never learn the foil items from the forced-choice test, since the transitional probability for each pair of shapes in a foil triple was 0. Thus, we decided it was best to investigate how well the target triple was learned, rather than looking at relative learning between the target triple and a foil triple that was impossible for two of the three models to have any sort of false confidence in.

Results. PARSER and MDLChunker both showed better learning of the target triple in the four-triple condition than in the one-triple condition (PARSER: $t(38) = 2.79$, $p = 0.008$; MDLChunker: $t(38) = 3.14$, $p = 0.003$). TRACX showed equivalent performance in both conditions, $t(38) = 0.05$, $p = 0.96$. Figure 3 shows the distribution of model outputs for each condition.

While both PARSER and MDLChunker matched the direction of the effect, PARSER's performance seems to match the experimental data better. MDLChunker learned the target triple in every single run of the model, though the average bit length was lower in the four-triple condition. PARSER showed greater variability: PARSER learned the target triple in only 11 of 20 runs in the one-triple condition, but in 19 of 20 runs in the four-triple condition. PARSER's occasional lack of learning maps onto the forced-choice data a bit more naturally than MDLChunker's varying degrees of learning. PARSER might genuinely predict uncertainty between the target and foil triple when the target is not learned, but MDLChunker always learned the target to some degree.

Discussion

We presented results from an experiment designed to investigate how the learning of a chunk is influenced by the presence or absence of other chunk-able information. We found that a chunk was better learned when it was embedded in a sequence that was also chunk-able than when it was embedded in a more randomly generated sequence. We tested three process models on this task, and found that two of them, PARSER and MDLChunker, predicted a difference in learning between the two conditions, while the third, TRACX, did not.

Why did PARSER and MDLChunker both match the direction of the effect, while TRACX showed equivalent performance in both conditions? The key difference seems to be the way that memory constraints are implemented in the models. PARSER and MDLChunker both share a common feature: As the models learn to chunk the input sequence, the relative strength of the memory encoding for individual chunks increases. In both models, this effectively leads to a longer lasting memory for previously seen chunks. The longer memory span improves learning for individual chunks, as they seem to be more frequent from the perspective of the memory-limited model. We'll illustrate this by walking through each model.

PARSER processes a sequence in sets of 1, 2, or 3 units at a time. The number of units is randomly selected at each model step. Consider the sequence `ABCGHIDEFABC`. If PARSER contains no chunks, and randomly selects to see 3 items, then the input on this step will be `A/B/C`. But, if PARSER has already learned the chunks `ABC`, `GHI`, and `DEF`, then the input would be `ABC/DEF/GHI`. In both cases, the chunk `ABC` will be reinforced, increasing its weight in memory. However, on the next step, the version with no chunks will see the input `G` (supposing that PARSER randomly chooses 1 unit as the input), and the `ABC` chunk will decay slightly in memory. The version with chunks will see `ABC` again, since it has already processed the first nine items in the sequence, reinforcing `ABC` even further. When PARSER is able to chunk the input sequence, it can process the input in fewer model steps, as shown by this toy example. This has the effect of accelerating the exposure rate of chunks. Since the decay rate of items in memory is fixed to the number of model steps, an individual chunk will experience less decay between successive presentations when the intermediate sequence is chunk-able. This process could equivalently be thought of as decreasing the decay rate of stored items when the incoming items are chunks. PARSER, in essence, behaves like it has a longer lasting memory when the input sequence is chunk-able than when it is not.

MDLChunker ends up with functionally similar behavior, but through a different kind of memory limitation. In MDLChunker, the minimum description length (MDL) is calculated on a set of two components: the set of chunks the model has stored in its lexicon, and the input sequence coded in terms of the chunks in the lexicon. The bit length of an individual chunk depends on the relative frequency of that chunk in memory. In the one-triple condition, the optimal encoding would be one triple and nine singletons, so the relative frequency of the triple will be, on average, 1/10. In the four-triple condition, the optimal encoding would be four triples, and the relative frequency of the target triple would be 1/4. Since the bit length of an individual chunk depends on its frequency in memory, the

bit length of the target chunk is smaller when the surrounding sequence also contains chunks. If we take bit length to indicate relative strength of encoding, then the target chunk will have a stronger encoding in the four-triple condition, due to an increase in relative frequency.

TRACX, in contrast, has no explicit memory storage nor any explicit forgetting parameter. TRACX also processes a sequence at a rate of one item per step regardless of previous learning. Memory constraints in TRACX will depend on interference in learning connection weights. Thus, TRACX lacks the kind of mechanism that we hypothesize might be responsible for the observed effect.

This interpretation of the model results makes a key prediction: The reinforcement schedule necessary for successful chunk learning depends on the complexity, defined in terms of the perceiver's internal model/representation, of the information that is seen in between successive presentations of the chunk. When the information between successive presentations of a chunk is highly compressible, then less frequent presentations are necessary to support chunk formation. However, when the information between sequences is unpredictable, then more frequent presentations of the chunk are necessary in order for learning to take place. This prediction can be tested empirically in future work.

Acknowledgments

This material is based on work that was supported by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1342962.

References

Aslin, R. N., Saffran, J. R., & Newport, E. L. (1998). Computation of Conditional Probability Statistics by 8-Month-Old Infants. *Psychological Science*, *9*(4), 321–324. doi:10.1111/1467-9280.00063

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, *61*, 55–61.

de Leeuw, J. R. (2015). jsPsych: A JavaScript library for creating behavioral experiments in a Web browser. *Behavior Research Methods*, *47*(1), 1–12. doi:10.3758/s13428-014-0458-y

Fiser, J., & Aslin, R. N. (2001). Unsupervised statistical learning of higher-order spatial structures from visual scenes. *Psychological Science*, *12*(6), 499–504.

Fiser, J., & Aslin, R. N. (2002). Statistical learning of higher-order temporal structure from visual shape sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *28*(3), 458–467. doi:10.1037/0278-7393.28.3.458

Frank, M. C., & Gibson, E. (2011). Overcoming Memory Limitations in Rule Learning. *Language Learning and Development*, *7*, 130–148. doi:10.1080/15475441.2010.512522

Frank, M. C., Goldwater, S., Griffiths, T. L., & Tenenbaum, J. B. (2010). Modeling human performance in statistical word segmentation. *Cognition*, *117*, 107–125. doi:10.1016/j.cognition.2010.07.005

French, R. M., Addyman, C., & Mareschal, D. (2011). TRACX: a recognition-based connectionist framework for sequence segmentation and chunk extraction. *Psychological Review*, *118*(4), 614–636. doi:10.1037/a0025255

Gebhart, A. L., Aslin, R. N., & Newport, E. L. (2009). Changing structures in midstream: Learning along the statistical garden path. *Cognitive Science*, *33*, 1087–1116. doi:10.1111/j.1551-6709.2009.01041.x

Gobet, F., Lane, P. C., Croker, S., Cheng, P. C. H., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, *5*(6), 236–243.

Gobet, F., & Simon, H. A. (1998). Expert chess memory: revisiting the chunking hypothesis. *Memory*, *6*(3), 225–255. doi:10.1080/741942359

Klahr, D., Chase, W. G., & Lovelace, E. A. (1983). Structure and process in alphabetic retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *9*(3), 462–477.

Kruschke, J. K. (2011). *Doing Bayesian Data Analysis: A Tutorial with R and BUGS* (1st ed.). Academic Press.

Lany, J., & Gómez, R. L. (2008). Twelve-month-old infants benefit from prior experience in statistical learning. *Psychological Science*, *19*(12), 1247–1252.

Lew-Williams, C., & Saffran, J. R. (2012). All words are not created equal: Expectations about word length guide infant statistical learning. *Cognition*, *122*, 241–246. doi:10.1016/j.cognition.2011.10.007

Perruchet, P., & Pacton, S. (2006). Implicit learning and statistical learning: one phenomenon, two approaches. *Trends in Cognitive Sciences*, *10*(5), 233–238. doi:10.1016/j.tics.2006.03.006

Perruchet, P., & Vinter, A. (1998). PARSER: A Model for Word Segmentation. *Journal of Memory and Language*, *39*, 246–263. doi:10.1006/jmla.1998.2576

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*(5), 465–471.

Robinet, V., Lemaire, B., & Gordon, M. B. (2011). *MDLChunker: A MDL-based cognitive model of inductive learning*. *Cognitive Science* (Vol. 35, pp. 1352–1389). doi:10.1111/j.1551-6709.2011.01188.x

Saffran, J. R., Aslin, R. N., & Newport, E. L. (1996). Statistical learning by 8-month-old infants. *Science*, *274*(5294), 1926–1928. doi:10.1126/science.274.5294.1926

Simon, H. A. (1974). How big is a chunk? *Science*, *183*, 482–488. doi:10.1126/science.183.4124.482