# Learning Via Compact Data Representation

**Mark W. Davis (madavis@crl.nmsu.edu)**
Computing Research Lab; New Mexico State University; Dept. 3CRL; Box 30001;
Las Cruces, NM 88003 USA

**Peter W. Foltz (pfoltz@crl.nmsu.edu)**
Department of Psychology, New Mexico State University; Dept. 3452; Box 30001;
Las Cruces, NM 88003 USA

**Abstract**

We present an unsupervised learning methodology derived from compact data encoding and demonstrate how to construct models of polysemy, priming, semantic disambiguation and learning using this theoretical basis. The model is capable of simulating human-like performance on artificial grammar learning.

## Introduction

William of Occam suggested a mechanism for deselecting models given nothing but examples of a process. Occam's suggestion, now known as Occam's Razor, posits that among the models that explain a phenomena, the simplest or most parsimonious should be chosen. Karl Popper rejected Occam's Razor as a methodological advance to inductive inference because he felt it was too subjective (Popper, 1959). Solomonoff (1964), however, again investigated inductive inference, or learning classificatory rules and models from examples, as a formal topic in information theory and statistics. The solution to Popper's quandary proposed by Solomonoff, and later developed by Rissanen (1978), was to consider the concept of simplicity as derived from informational code-length. Under this formulation, the statistical properties of an example data stream are used to estimate the properties of a communication model that can be used to transmit the shortest possible message. In so doing, the communicator must encode as much redundancy in the data as possible in their message. The subjectivity of Popper is replaced by an objective measure of information content derived from the statistical properties of the data and proposed model. The end result is that the minimum encoding of the data stream results in maximum accuracy of the predictions of the model.

In this paper, the consequences of the code-length formulation of inductive learning are investigated. A model is presented that effectively accounts for the acquisition of the relationships between terms with multiple, distinct meanings (polysemy) using this approach to compact data coding. Further, the same technique is shown to model human performance on artificial grammar learning and string recall experiments, and to explain the way in which meanings are observed to be activated during reading (lexical priming) (*e.g.*, McNamara, 1992).

## Bayes' Theorem and Compact Coding

We can formulate the problem of model selection for data as an extension of Bayes' Theorem. If we consider Bayes' Theorem operating on data $D$ and statistical models $M$

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)} \qquad (1)$$

we are then interested in comparing the posterior probability of different models given the data. If we constrain our models to just a specific class of models, then we are interested doing the same comparison over the model parameterizations within that class:

$$\arg\max_M P(M|D) = \arg\max_M \frac{P(D|M)P(M)}{P(D)} \qquad (2)$$

where $M$ is the model class. Applying -log to both sides, and dropping the *a priori* distribution of the data (the prior data distribution is constant over models) yields

$$\arg\min_M [-\log P(D|M) - \log P(M)] \qquad (3)$$

which says that we can choose the best model for a given set of data by choosing the most probable model $P(M)$ that also encodes the data in the most probable manner $P(D|M)$. Note that this is simply the maximum *a posteriori* formula in the space of models and data. By using the base-2 logarithm, this same formula can also be interpreted as meaning that a data model costs $|M|$ bits to encode (where $|M|$ is the bit length of the model) plus additional bits to specify the data using this encoding. The two interpretations are really only equivalent if the *a priori* model distribution is the *universal prior* on the model length,

$$P(M) = 2^{-|M|}, \qquad (4)$$

but the relationship is arguably justifiable (Li and Vitanyi, 1997, but also see Baxter and Oliver, 1995), or at least defensible as a heuristic (de Marcken, 1996). The additional cost to encode the data using that model is $-\log P(D|M)$. We then want to choose the model that minimizes the combined coding costs of data and model.

Rissanen (1978) characterized this approach as the Min-

imum Description Length (MDL) principle. MDL has been applied in pruning decision tree models (Quinlan and Rivest, 1989), grammar induction (de Marcken, 1996) and neural network training (Hinton, *et. al.*, 1995), while the subclass of maximum entropy models has found wide application in image reconstruction and data analysis (Smith and Grandy, 1982).

## Parsimony Networks and Word Meaning

A Parsimony Network (PN) is an application of compact coding that can be used to model the causative and correlative relationships among data. A PN is represented as a hierarchical network with the hierarchy grown using a concatenative model. The specific application that we will put PNs to in this section is the problem of learning polysemous term relationships from a collection of documents. The PN is self-organizing, starting with only the most basic syntactic elements and growing hierarchical structure to account for co-occurrence patterns in the data.

As a primary example, let us imagine that we have a collection of 36 documents that contain varied distributions of the terms money (M), bank (B), river (R) and finance (F) as given by the following table of term co-occurrences (entry $m_{ij}$ is the number of documents in which terms $w_i$ and $w_j$ both occur).

Table 1: Document co-occurrence counts for terms

| | M | B | F | R |
|---|---|---|---|---|
| M | | 25 | 10 | 1 |
| B | | | 0 | 10 |
| F | | | | 2 |

Note that although F and B do not explicitly co-occur, we would hope that the PN procedure would find a connection between them because of the chain of connections from M/F and M/B. Further, the procedure should automatically determine that the M/R and R/F co-occurrences are spurious.

We can build a parsimony network by considering the coding problem as follows. Suppose we want to communicate the documents to someone. The simplest encoding would be to transmit a document start symbol followed by codewords for each word in the document, then the next document start symbol, and so forth. If we assume we do not care about word order, however, pairs and groups of terms co-occur significantly enough that we may want to group them together in our codebook and assign them a unique codeword. That way the entire group of terms could be communicated by a single codeword. We only want to do this grouping, however, when the length of the new entry in our codebook is more than offset by the savings we achieve in transmitting the data using this modified codebook.

To build the PN, we must use a heuristic to search the space of concatenative models that operate on the co-occurrence matrix. We can guess that a relatively high frequency of co-occurrence should lead to an improved encoding, although we must use a generate-and-test style heuristic to examine the space of potential codeword merges to determine a (locally) optimal model. First consider the incremental progression from a codebook containing only codewords for documents, followed by the progressive addition of term merges,

U

U+BM

U+BM+BR

U+BM+BR+FM

U+BM+BR+FM+(BM)(FM)

where U is the lexicon populated only by the single elements B, M, R and F. The costs for performing these merges are given in the following table, where the bit cost of lexical entries and data are inversely proportional to the (negative) log probability of a symbol in the complete message:

Table 2: Description lengths of lexicon, data, and combined lexicon and data for merge operations.

| | Lexicon (bits) | Data (bits) | Total (bits) |
|---|---|---|---|
| U | 8.76 | 175.39 | 184.14 |
| U+BM | 13.16 | 163.56 | 176.72 |
| U+BM+BR | 21.38 | 146.10 | 167.48 |
| U+BM+BR+FM | 34.74 | 130.43 | 165.16 |
| U+BM+BR+FM+ (BM)(FM) | 45.67 | 125.17 | 170.84 |

The final entry includes a merge between two merged entities, and is not successful because the merge of the BM and FM entities does not result in a decrease in the overall description length of the data.

Given the set of merges in Table 2, the PN can be constructed by choosing the minimum total description length model seen thus far (U+BM+BR+FM) and constructing nodes for each entity in the lexicon, with connections between each node that is subsumed by that entity. The document codewords, although considered constants for the previous argument, may also be considered as part of the lexicon. The graph is constructed in a bottom-up fashion, with all of the document words serving as leaf nodes at the bottom. The first-level merges are then added, with connections to the words that they encode. This continues until the parse tree of the lexicon is fully represented. A document layer can then be added with connections to all of the lexical entities needed to encode the data. The complete PN for this data is shown in Figure 1.

Further, we can associate a weight with each connection in the network by estimating the conditional probabilities that connected nodes occur in the same document. For example, for the edge connecting BANK to RIVER-BANK, we would calculate a weight that is equal to

$$P(RB|B) = \frac{N_{RB}}{N_B} \qquad (5)$$

where $N_{RB}$ is the number of documents that contain RIVER and BANK (10), and $N_B$ is the number of documents that contain BANK alone (35). We treat the network as a Markov Network, similar to a Bayesian Network but without the restriction that the representation be a DAG (directed acyclic graph). Instead, we will be satisfied if we can capture correlational relationships as opposed to those that are definitively causal. Using the Markov Network, we can calculate the marginal probabilities of documents, merge-nodes and leaf terms given evidence of one or more leaf terms. For example, the activation of BANK with $P(B)$ will cause updates throughout the network, starting with the RB node:

$$P(RB) = P(RB|B)P(B) + P(RB|R)P(R) \qquad (6)$$

and the MB node:

$$P(MB) = P(MB|B)P(B) + P(MB|M)P(M) \qquad (7)$$

which, in turn activate documents with $P(D_i|RB)$ and $P(D_i|MB)$, respectively. Further, we can propagate the effect of a BANK activation through RB and back to RIVER using:

$$P(R) = P(R|RB)P(RB|B)P(B) \qquad (8)$$

So, for our example, an activation of BANK with $P(B) = 1$ would lead to $P(RB) = 10/35 = 0.2857$ and RIVER would, in turn, be activated at that same strength since $P(R|RB) = 1$.

Now, if we activated MONEY, $P(M) = 1/2$, and BANK, $P(B) = 1/2$, at the same time, the spread of activations to the MB node would be given by

$$P(MB) = P(MB|B)P(B) + P(MB|M)P(M)$$
$$P(MB) = \frac{1}{2}\left(\frac{N_{MB}}{N_B} + \frac{N_{MB}}{N_M}\right) = \frac{25}{70} + \frac{36}{72} = 0.7044 \qquad (9)$$

We can also see that the RB and FM nodes will be activated to a lesser degree than the MB node:

$$P(RB) = (10/35)(1/2) = 0.1428 \qquad (10)$$

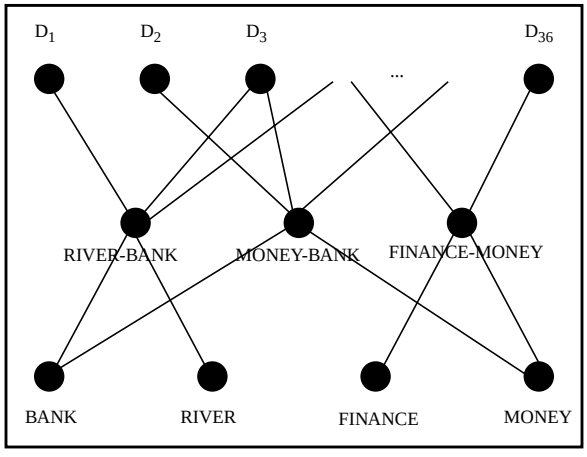


Figure 1: Parsimony Network (PN) constructed from the minimum description length merge data.

$$P(FM) = (10/35)(1/2) = 0.1428 \qquad (11)$$

The PN algorithm thus shows the primary properties that we set out demonstrate; it provides an algorithm for self-organization of correlational data with a built-in criterion for determining how the self-organization should proceed. The algorithm provides an automatic method for, in effect, assigning a hierarchy of categories to co-occurring events based only on intrinsic structure of the data. Further, the correlational data, as expressed in the parsimony network, expresses latent relationships among variables that share significant correlations with a given event, but which do not, themselves, co-occur.

This last point emphasizes the similarity of the procedure to Latent Semantic Analysis, which has been used to model synonymy and human performance on learning tasks (*e.g.*, Landauer and Dumais, 1997; Landauer, Laham and Foltz, 1997). In Latent Semantic Analysis, a factor-analytic approach to data analysis, a singular value deconstruction (SVD) is performed on a matrix of feature co-occurrence patterns—often term-document counts or a related statistic. A direct comparison can be made between the reconstructed version of the SVD of the preceeding term by term matrix, and the matrix built by the PN algorithm:

$$M = \begin{bmatrix} 1 & \frac{N_{MB}}{N_M} & \frac{N_{MF}}{N_M} & \frac{N_{MB}N_{RB}}{N_M N_B} \\ 0 & 1 & \frac{N_{FM}N_{MB}}{N_M N_B} & \frac{N_{RB}}{N_B} \\ 0 & 0 & 1 & \frac{N_{RB}N_{FM}N_{MB}}{N_M N_B N_F} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (12)$$

This matrix is highly correlated with the matrix produced by Latent Semantic Analysis (first singular value) for the same data, $r = 0.897$. This close relationship to factor analysis has been examined before. Wallace and Freeman (1992), for example, propose a proof that MML (their formulation of the MDL principle) can be used for single factor analysis.

## Parsimony and Priming

While the analytic and empirical features of the PN procedure appear useful, we can also see in the networks an isomorphism to connectionist architectures. If we assume that the speed of propagation through the PN is isomorphic to the speed of cognitive access to a stored lexicon, activation of a term like BANK will, in one step, spread to both connected senses of the term, RIVER-BANK and MONEY-BANK, with activation levels conditioned on the base rates of their co-occurrences in the document space. If MONEY then occurs, the MONEY-BANK node will receive additional activation, while the RIVER-BANK node will receive extremely weak activation through a three step activation sequence later on (M->MB->B->RB). By choosing the node that has the highest probability at each stage the PN network disambiguates terminology in a manner that fits with the lex-

ical priming results. Specifically, the network behavior is consistent with the findings that all senses of a term appear to be activated within an initial 300-500 ms window after exposure to a prime, but that activation quickly settles on the correct sense of a term after the initial 300-500 ms window (*e.g.*, McNamara, 1992; Kintsch, 1988). Further, frequency effects that have been observed in priming experiments are explicitly handled due to the differential weighting of the PN edges by their marginal frequencies.

## Learning Through Parsimonious Integration

PNs provide a mechanism for the acquisition of structure from a complete database, but there is no reason to suspect that the same mechanism translates directly into an on-line procedure for learning from partial data in a way that corresponds to observations of human learning. In this section, an alternative version of the PN algorithm is developed that can be used for on-line learning. The algorithm has a retrospective component that may reorganize the PN as new evidence presents itself so the algorithm can be considered integrative. The procedure, which we call Parsimonious Integration (PI), is as follows:

1. Maintain a lexicon that contains a PN-like hierarchy of merged, observed events.
2. Observe—and re-examine if possible—all available events, merging together any events and encoded event groups that appear with frequency >1.
3. Use local calculations to heuristically reorganize the lexicon. Attempt to reorganize lexicon subtrees if an observed pattern in the lexicon spans multiple elements. Do the reorganization based on the comparative encoding costs of the two alternative subtrees.

The procedure can be seen as similar to Nevill-Manning's SEQUITUR algorithm (Nevill-Manning, 1996), but with a global re-evaluation component that undoes grammatical changes based on global observations, yet which is simpler than the complex global observations in the PN construction procedure of the previous sections. The Parsimonious Integration procedure makes partial use of downstream calculations to attempt to reorganize the constructed grammar as new information becomes available. The following example in artificial grammar induction illustrates the PN procedure.

### Artificial Grammar Learning

As an example of how PNs can be learned, we will examine data from an artificial grammar. The data strings will be generated from a stochastic finite-state machine that corresponds to a grammar used by Vokey and Brooks (1992) in artificial grammar implicit learning studies. The state machine, as well as some strings consistent with the grammar, are shown in Figure 2.

For this experiment, random sequences of consistent strings were produced by an algorithm that initially began at node 1, then transited to state 2 or 3 based on a random coin

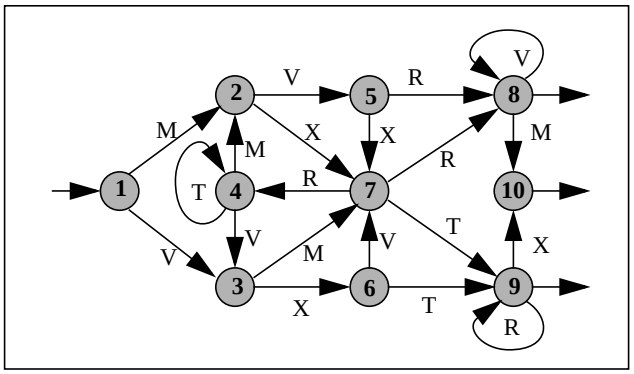


Figure 2: Finite-state machine (from Vokey and Brooks, 1992).

flip, and so forth, stochastically working through the states until exiting from states 8, 9 or 10. The first 10 such strings are shown below:

MXRVVM

VXTX

VXT

VXVTX

VXVTRX

VXTR

MVRM

MVRM

MVXT

VMTX

If we incrementally apply the Parsimonious Integration algorithm above to these strings, we first recognize the VV in the first string, so our lexicon contains VV and we can represent the encoding of the string with MXR(VV)M. With the second and third strings, we discover the commonality VXT and add it to our lexicon. But then by committing to the VXT encoding, when we encounter the fourth string VXVTX, we must encode it as (V)(X)(V)(T)(X) and can't take advantage of the shared VX pattern with the previous strings. We apply a heuristic-version of the PN procedure at this point, counting the number of lexicon and data elements produced by two alternate formulations. These calculations are conducted locally with respect to the strings VXTX, VXT, VXVTX (i.e. we do not modify the probabilities of other strings in the complete lexicon based on these changes). So, the encoding (VXT)(X), (VXT), (V)(X)(V)(T)(X) with the lexicon (VXT) requires a lexicon of size 1 to create data of size 8. A reformulation would be to add VX and VXT to the lexicon and code the data ((VX)T)(X), ((VX)T) and (VX)(V)(T)(X), resulting in a lexicon of size 2, data of size 7. At this point the value of creating the additional VX lexical element appears to be

equivalent to the value of coding things separately because we don't have additional examples of the VX? pattern, where the ? represents any character except T. If we did have another example, it would tilt the evidence in favor of the recoding of the lexicon to include the VX element.

For the purposes of the Parsimonious Integration algorithm, the criterion under which we decide whether to evaluate a recoding is based on observing a new pattern that contains a substring found in an existing lexical element, or that spans two lexical elements, but which is not already in the lexicon. Note that the PI algorithm bears a similarity to the Competitive Chunking (CC) model suggested by Servan-Schreiber and Anderson (1990), but with the use of the MDL heuristic mechanism for incorporating the observed chunks into a hierarchical model. The CC model uses a concept of support that is closely related, but not identical to, the MDL heuristic incorporated into the PI algorithm.

Most experiments to test implicit learning of artificial grammars involve subjects who have studied productions from the grammar, then must decide whether new strings are from that grammar or not. The test set must be carefully constructed so that the strings are balanced in terms of the frequencies of repeated bigrams and trigrams within the strings of the test versus training sets, otherwise the subjects may simply be deciding on grammaticality based on short, repeated sequences (Higham, 1997). As a model, the PNs constructed by the Parsimonious Integration procedure can be used to decide on grammaticality by parsing observed sequences with the constructed PN lexicons, then choosing grammaticality based on the coverage of the string afforded by the PN lexicon. For example, given the lexicon built from the first four strings in the last example, if we observed the string VXVR, we could parse it as (VX)(V)(R), resulting in a 50% chance of choosing this as a legal production from the grammar given only the information of the first four strings. Following Servan-Schreiber and Anderson's (1990) suggestion, the scoring of a string parse should be inversely proportional to the number of chunks needed to represent the string. The suggestion is that the subject's degree of confidence in a given string being grammatical increases up to the point at which it can be encoded as one chunk. One way to do this is to assign the probabilities according to a decreasing function of the number of chunks needed to code the string:

$$P(S) = 1 - \frac{N}{|S|} \qquad (13)$$

where $N$ is the number of chunks needed to parse the string. Note that we need to consider the unigrams in this formulation, so (VX)(V)(R) is parsed into 3 chunks, giving a 25% chance of grammaticality, (VXV)(R) has a 50% chance and (V)(X)(V)(R) has a 0% chance.

Artificial grammar induction experiments typically involve subjects who first examine sets of strings created by the production system. In Higham (1997), for example, 16 training strings were first presented to test subjects in random order for 3s each. Subjects were also asked to reproduce the training strings on a piece of paper. After observing all the training strings, the participants were told that the strings had been generated according to a rule system, although they were not told the nature of the rule structure. They were then tested on other strings that did not occur in the training set and asked to decide whether the strings were grammatical according to the rules system used in creating the training strings. Seven sample training strings, as well as a subset of the test strings, used in Higham (1997) are reprinted in Table 3.

Table 3: Training and test strings from Higham (1997)

| Training strings | Test strings | |
|---|---|---|
| | Grammatical | Nongrammatical |
| MXRVXT | MXRMXT | MTR |
| MXTRRR | VXTRRR | VXVRTXT |
| VXVRMXT | VXVRVV | VMRVVVR |
| VXVRVM | MXRTMXR | VMRTXTR |
| MXRTMVR | VMTRRRX | VMRMTRV |
| VMRMXTR | MXRVM | MMRMVRM |
| MXR | VMRVVVM | MXRTRXT |

The PI algorithm can be applied directly to the training strings shown in Table 3. The PI-constructed lexicon is shown in Table 4, below.

Table 4: PI-constructed lexicon for artificial grammar training.

| | | |
|---|---|---|
| MX | XT | RR |
| (MX)R | M(XT) | (VX)T |
| MV | (M(XT))R | VR |
| (MV)R | TR | (VX)(VR) |
| VM | (TR)(RR) | (VR)(VM) |
| (VM)R | VX | VV |

Using this grammar, we can perform left-to-right greedy parses of the test strings and calculate the coding probability using (13). The first two parses and their corresponding probabilities are shown in Table 5, below.

Table 5: Parses and recognition probabilities for grammatical and nongrammatical test strings

| **Grammatical** | |
|---|---|
| **Parse** | ***P(S)*** |
| ((MX)R)(M(XT)) | 0.67 |
| ((VX)T)(RR)R | 0.50 |
| **Nongrammatical** | |
| **Parse** | ***P(S)*** |
| MTR | 0.0 |
| ((VX)(VR))T(XT) | 0.57 |

Overall the mean performance figures were 0.58 ($\sigma = 0.064$) for the grammatical strings and 0.39 ($\sigma = 0.222$) for the nongrammatical. A t-test shows that the differences are significant at $p = 0.05$. Human subjects also show significant differences for this data set, with human

subjects also able to detect grammaticality even given carefully balanced data sets (Higham, 1997).

## Conclusions

The philosophical position of William of Occam suggests that simple models lead to better models. Yet the apparent subjectivity of the notion of simplicity has until fairly recently left Occam's Razor without more than a dull, retrospective edge. With careful formulation of heuristics based on information-theoretic grounds, however, we have demonstrated how Occam's suggestion can become the methodological backbone of a theory of inductive inference with practical—and testable—consequences for understanding and modeling human intelligence.

The model is unique among related psychological theories in that it presents a fully-automated induction method (with minimal arbitrary parameters) for learning predictive models from data in a completely unsupervised fashion. In our current work, we are interested both in practical applications, like using the heuristic version of the Parsimonious Integration model for information retrieval, and in further understanding the model's capacity for simulating psychological processes. For example, although a promising model for artificial grammar learning, more work is needed to understand how implicit versus explicit knowledge can be represented in hierarchical compact representations. We are currently modeling human memory data originally obtained by Miller (1958) in which subjects are asked to recall random and redundant strings. We have discovered that the probability of recalling random strings is significantly lower than recalling redundant ones using the PN algorithm. Overall, the wide range of problems that appear admissible to compact data encoding suggests this method has significant implications for understanding cognition.

## Acknowledgments

The authors are indebted to Ted Dunning, Bill Ogden and Wirt Atmar for constructive comments, input and support during the development and organic revision of this paper.

## References

Baxter, R. A., & Oliver, J. J. (1995). *MDL and MML: Similarities and Differences.* (Tech Report 207). Melbourne, Australia: Monash University, Department of Computer Science.

Higham, P. A. (1997). Dissociations of grammaticality and specific similarity effects in artificial grammar learning. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 23, 1-17**.**

Hinton, G. E., Dayan, P., Frey, B. J., & Neal, R. M. (1995) The wake-sleep algorithm for unsupervised neural networks. *Science*, 268, 1158-1161.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: a construction-integration model. *Psychological Review*, 95, 163-182.

Landauer, T.K., & Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104, 211-240

Landauer, T. K., Laham, D., & Foltz, P.W. (1997). Learning human-like knowledge with singular value decomposition: a progress report. *Proceedings of Neural Information Processing Systems, NIPS-97*.

Li, M., & Vitanyi, P. (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, NY.

de Marcken, C. (1996). *Unsupervised Language Acquisition*. Doctoral Dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

McNamara, T. P. (1992). Theories of priming: I. Associative distance and lag. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 18, 1173-1190.

Miller, G. A. (1958). Free recall of redundant strings of letters. *Journal of Experimental Psychology*, 56, 485-491.

Nevill-Manning, C. G. (1996). *Inferring Sequential Structure*. Doctoral Dissertation, Department of Computer Science, University of Waikato, New Zealand.

Popper, K. R. (1959). *The Logic of Scientific Discovery.* Toronto: University of Toronto Press.

Quinlan, J. R., & Rivest, R.L. (1989). Inferring decision trees using the minimum decision length principle. *Information and Computation*, 80, 227-248.

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

Servan-Schreiber, E., & Anderson, J. R. (1990). Learning grammars with competitive chunking. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 16, 592-608.

Smith, C. R., & Grandy, W. T. (1982). *Maximum-entropy and Bayesian methods in inverse problems.* D. Reidel Pub. Co., Boston.

Solomonoff, R. J. (1964). A formal theory of inductive inference. (part 1 and part 2). *Information and Control*, 7, 1-22, 224-254.

Vokey, J. R., & Brooks, L. R. (1992). Fragmentary knowledge and processing-specific control of structural sensitivity. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 20, 1504-1510**.**

Wallace, C. S., & Freeman, J. R. (1992). Single factor analysis by minimum message length. *J. Royal Stat. Society. B.*, 54, 195-209.