What's in a chunk? Chunking and data compression in verbal short-term memory.

Dennis Norris

and

Kristjan Kalm

MRC Cognition and Brain Sciences Unit,

University of Cambridge, Cambridge, UK

Contact Information:
dennis.norris@mrc-cbu.cam.ac.uk

Dennis Norris
Medical Research Council Cognition and Brain Sciences Unit,
University of Cambridge,
15 Chaucer Road,
Cambridge, CB2 2EF,
UK

Abstract

Short-term verbal memory is improved when words in the input can be chunked into larger units. Miller (1956) suggested that the capacity of verbal short-term memory is determined by the number of chunks that can be stored in memory, and not by the number of items or the amount of information. But how does the improvement due to chunking come about? Is memory really determined by the number of chunks? One possibility is that chunking is a form of data compression. Chunking allows more information to be stored in the available capacity. An alternative is that chunking operates primarily by redintegration. Chunks exist only in long-term memory, and enable items in the input which correspond to chunks to be reconstructed more reliably from a degraded trace. We review the data favoring each of these views and discuss the implications of treating chunking as data compression. Contrary to Miller, we suggest that memory capacity is primarily determined by the amount of information that can be stored. However, given the limitations on the representations that can be stored in verbal short-term memory, chunking can sometimes allow the information capacity of short-term memory to be exploited more efficiently. (196 words)

Keywords: memory, short-term memory, chunking

Chunking in verbal short-term memory

The idea that verbal STM capacity is strongly influenced by the number of chunks that can be held in STM has become part of the conventional wisdom. Miller's (1956) famous work on chunking "The magical number seven" argued that the amount of material that can be stored in short-term memory (STM) is not determined by limits on the number of items nor by the amount of information, but by the number of chunks. However, one important question has received very little attention: What is a chunk?

Let us assume that there is some sense in which FBI is a chunk, and that this confers a memory advantage over an unfamiliar sequence of letters. What are the properties of the representation of that chunk that lead to superior memory? STM must represent something other than the raw sequence of letters in the chunk, so what might that be? Miller himself acknowledged that "we are not very definite about what constitutes a chunk of information." (p93). One simple suggestion comes from Shiffrin and Nosofsky (1994) who defined a chunk as ''a pronounceable label that may be cycled within short-term memory''. They went on to point out that "The label must have the critical property that it can be unpacked accurately into its original constituents." (p 360). Similarly, Miller suggested that there are many ways to perform chunking "but probably the simplest 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." (p93). That is, chunking is achieved by *recoding* chunks in the input into a different vocabulary. Bower (1970) made a similar suggestion to account for chunking in LTM. The idea of chunks as verbal labels would seem to apply most directly to those instances where memory is increased by an explicit recoding strategy. For example, Miller reported a study by Smith. Smith taught himself to chunk binary digits by recoding them

into octal digits. Three binary digits can be recoded as a single octal digit. Here the octal digit becomes the label for the chunk of thee binary digits. Smith could recall about 12 octal digits and, by using the recoding method, he could recall about 36 binary digits. Even more elaborate chunking schemes have been reported by Ericcson, Chase, and Faloon (1980). After extensive training, their participant SF was able to recall up to 79 digits. SF made extensive use of his knowledge of running times to recode the digits. Training did not increase his STM capacity. Despite extensive practice with digits, SF did not increase his memory for consonants. The improvement appeared to be entirely attributable to the development of efficient chunking strategies.

In this paper we begin by reviewing existing suggestions about the nature of chunks in STM. The simplest proposals assume that chunking is achieved by recoding. Recoding schemes are examples of data compression, and capitalize on redundancy in the input to allow more items to be stored in the available capacity. We provide an extensive discussion of data compression and its implications for models of STM. The distinctive feature of data compression schemes is that compression must change the contents of STM. The compressed code must replace the original message in STM. An alternative view is that the benefits of chunking might arise from what is often termed redintegration. According to this view, chunking does not change the contents of STM at all. Chunks exist only in LTM, and LTM representations help recover degraded information form STM. The evidence suggests that both compression and redintegration can play a role in STM, but that it is possible to differentiate between them.

We conclude by offering a revised account of chunking. We suggest that chunking depends on the nature of the representational vocabulary underlying a particular memory

system and the degree of redundancy in the input. Only sometimes will this give the impression that capacity is determined by chunks.

Chunking and grouping

Following Miller, we will make a distinction between chunking and grouping. Whereas chunks are determined by pre-existing representations in LTM, grouping is determined by the perceptual structure of the input. For example, in serial recall, lists of digits might be grouped in threes by inserting pauses between the digits, and this will enhance recall (Broadbent & Broadbent, 1981; Frankish, 1985; Henson, Burgess, & Frith, 2000; Hitch, Burgess, Towse, & Culpin, 1996; Ryan, 1969a, 1969b; Wickelgren, 1964), but those groups need not correspond to any existing representations in LTM. In contrast, a list of letters such as IBMBBCFBI is an ungrouped list consisting of three chunks: IBM, BBC, FBI.

However, grouping can influence the formation of chunks. For example, McLean and Gregg (1967) had participants learn lists of 24 letters presented in groups of between one and eight letters. They reported that the timing of pauses in spoken recall reflected the pattern of grouping, suggesting that the groups had become chunks in LTM. Bower and Winzenz (1969) and Winzenz (1972) found that learning of repeated lists of digits in a Hebb (1961) repetition paradigm was impaired when the grouping of the digits changed from trial to trial. Similarly, Schwartz and Bryden (1971) found that if the initial items in a list varied across repetitions, there was no learning, whereas there was learning if the final items varied. It seems that in order to learn a chunk, the onset of the chunk must be marked by a grouping boundary, such as the beginning of a list. In all three of these studies, grouping determines which items become learned as chunks in LTM.

Terminology

In the present context there are at least three things to which ‘chunk’ might refer. The representation of FBI in LTM might be called a chunk, and so might the representation that codes the chunk in STM. A chunk in STM might be coded by a label, or perhaps by a pointer to the representation in LTM. It is also possible that the representation of a chunk in STM might have the same form as the representation of the chunk in LTM. This is the case with redintegration theories that we will discuss later. The literature rarely pays heed to these differences. Usually the term chunk appears to be used to refer to the combination of the representations of the chunk in LTM and STM. When discussing chunking in general terms we will use ‘chunk’ in that theory-neutral sense. However, given that our aim is to specify the nature of the chunking process in more detail, when presenting our theoretical analysis we will try to be more specific.

Although our concern here will be with the influence of chunking on STM, the concept of chunking appears in different guises in many areas of cognition. Interestingly, in a recent review of the meanings of chunks and chunking (Gobet, Lloyd-Kelly, & Lane, 2016) there is no mention of chunking in STM. Many models of learning assume that learning involves creating chunks that can be combined to form a hierarchy of chunks (e.g. ACT-R: Anderson, 1983; CHREST: Gobet, 1993; CLASSIC: G. Jones, Gobet, Freudenthal, Watson, & Pine, 2014; SOAR: Newell, 1994; MDL Chunker: Robinet, Lemaire, & Gordon, 2011; EPAM: Simon & Feigenbaum, 1964). Gobet et al. (2001) suggested that it was possible to produce a common definition of a chunk as “a collection of elements having strong associations with one another, but weak associations with elements within other chunks” (c.f. Chase & Simon, 1973; Cowan, 2001; Simon, 1974). There are other more elaborate notions of chunking. For

example, in ACT-R (Anderson, 1983; Anderson, Bothell, Lebiere, & Matessa, 1998) chunks are viewed as schema-like structures containing pointers which encode their contents. In the present context these approaches to the study of chunking can be seen as investigations into the structure of chunk representations in LTM. The focus here is on how those representations influence storage in STM.

Does verbal STM have a constant chunk capacity?

Miller argued that the capacity of STM was not determined by the amount of information that could be held in the store. Nowadays we standardly think of computer storage in terms of information capacity – the number of bits or bytes. How did Miller come to the conclusion that STM capacity was not a function of the amount of information that could be stored? Miller based his argument on data from studies by Hayes (1952) and Pollack (1953) which compared memory span for different types of materials. If memory span were determined by the amount of information required to store the items, then items selected from a large set should take up more capacity than items selected form a small set. In a message that is known to consist only of decimal digits, the identity of each digit requires 3.3 bits of information to transmit. One of the 26 letters of the alphabet requires 4.7 bits, and one word form a set of 1000 words requires 9.97 bits. A system with a capacity of about 10 bits should therefore be able to store approximately one word, two letters of the alphabet, three decimal digits, or 10 binary digits. Span should therefore be lower for decimal digits than for binary digits, lower for letters, and lower still for words. Hayes measured memory span for lists of binary digits, decimal digits, letters, letter plus digits, and words from a set of 1000. However, Hayes found that span, in terms of number of items

recalled, was roughly equivalent for all classes of stimulus. A similar result was reported by Pollack. Miller, Hayes and Pollack all took this to imply that the limiting factor on STM storage was not information capacity as measured in bits. From the chunking study by Smith, and many similar studies, we know that span can be increased by chunking. The conclusion therefore was that capacity is not determined by the number of individual items that can be stored, but by the number of chunks into which the message can be recoded. However, as we will see later, this conclusion depends on the assumption that people know the set of items that they will be tested on and that they have developed an optimal code for representing specifically that set items. This seems highly unlikely.

The most thorough investigation of chunking in verbal STM comes from a series of studies by Cowan and colleagues (Chen & Cowan, 2005, 2009; Cowan & Chen, 2008; Cowan, Chen, & Rouder, 2004; Cowan, Rouder, Blume, & Saults, 2012). The main aim of these studies has been to establish whether memory capacity is limited by the number of items that can be held in memory or by the number of chunks. They had their participants learn chunks comprised of different numbers of words. They were then tested on either free recall, serial recall, or forced-choice recognition of lists containing different numbers of chunks. For example, Chen and Cowan (2009) familiarized their participants with pairs of words (e.g. *brick-hat, king-desk*) and with single words in the context of a cued recall task. Singletons and pairs of words and were both considered to comprise a single chunk. The words were presented repeatedly until participants' performance was perfect. This familiarization phase was followed by a serial recall task using lists of 2, 4, 6, 8 or 12 singletons, or 4 or 6 learned pairs. Participants performed articulatory suppression during list presentation and then typed their responses. Responses were scored in terms of the

number of chunks (singletons or pairs) that were recalled in either the correct position (strict scoring) or anywhere in the list (lenient scoring).

Under the lenient scoring procedure the same number of chunks was recalled regardless of the number of words in the lists. In other words, more items were recalled when the lists contained more pairs. Chen and Cowan took this as evidence of a chunk-based capacity limit on verbal working memory. However, with strict scoring the number of chunks recalled was also influenced by the number of items in the list. For example, more chunks were recalled from lists of four singletons than from lists of four pairs. As both of these lists contain four chunks, performance should be equivalent when measured in terms of chunks recalled. Also the chunk-based limit only appeared to hold under conditions combining articulatory suppression with lenient scoring.

Chen and Cowan (2005) had earlier found little evidence of a chunk-based limit when no suppression was required. In fact, for eight-item lists there was no difference between lists consisting of either eight singles or four pairs. This led Chen and Cowan (2009) to conclude that "There is no apparent chunk-based constant capacity for strict serial recall." (p1425)

Cowan, Rouder, Blume and Saults (2012) extended this work using a greater range of chunk sizes and list lengths. Their results qualify those of Chen and Cowan (2009) in two important respects. First, as chunks are made larger (contain more words) performance is poorer than predicted by constant chunk capacity. That is, performance tends to decline as more items must be recalled, even if the number of chunks remains constant. Second, when lists are made longer by increasing the number of singletons, performance is better than

would be expected on the basis of constant chunk capacity. Qualitatively similar observations were reported by Simon (1974), albeit using only himself as a subject.

Cowan et al. fitted a number of mathematical models to their data. The best-fitting model (Model VIII) added two components to the simple constant capacity model. The first allowed for the possibility of chunk decomposition; some chunks may break down into their components and no longer function as intact chunks. This accounts for the decline in performance with larger chunks. The second allowed for the possibility that some chunks may be stored in activated long-term memory which was assumed to have unlimited capacity. This accommodates the finding that chunk recall does not decline as much as expected when more singletons are incorporated into the lists. Note that none of the models they investigated has anything to say about the detailed representation of chunks. Chunks are simply representations in LTM from which the input can potentially be reconstructed.

A more specific suggestion can be found in earlier work by Chen and Cowan (2009). They suggested that "An advantage of acquiring multiword chunks is that it is then unnecessary to keep each word in capacity-limited working memory, but just some index to each chunk, perhaps the chunk's first word." (p 1428). According to this suggestion, the label for the chunk is simply its first word. We might call this a *deletion code.* With a deletion code the vocabulary of the message remains unaltered. This suggestion also acknowledges the nature of the problem; the representation of the chunk in STM must be different from the representation of the contents of the chunk itself and, in order to improve memory capacity, it must be shorter than the chunk. Furthermore, there has to be a clear mapping between STM and LTM such that the chunk can be reconstructed from whatever code is

used in STM. Interestingly, Chen and Cowan (2009), and Cowan, Rouder, Blume and Saults (2012) had their participants perform articulatory suppression during the presentation of the lists and still found evidence of chunking. This seems contrary to Shiffrin and Nosofsky's suggestion that the label should be pronounceable, as articulatory suppression would be expected to prevent a *pronounceable* label from being recycled within short-term memory. Perhaps it is sufficient to have a shorter verbal chunk label, regardless of whether it can be recycled. Overall, and contrary to Miller's suggestion, there is little evidence that memory capacity determined solely by the number of chunks that can be stored.

Data Compression

An intuitive notion of chunking is that it enables us to squeeze more information into a limited memory capacity. That is, chunking is an example of data compression (Brady, Konkle, & Alvarez, 2009; Chekaf, Cowan, & Mathy, 2016; Huang & Awh, 2018; Mathy & Feldman, 2012). Data compression is only possible when there is redundancy in the signal. For example, in the experiments of Cowan et al, subjects learned multi-word chunks such as *brick-hat*. Assuming that neither *brick* nor *hat* ever appears alone in the experiment, the chunk contains redundant information. If the list contains *brick*, the next word must be *hat*. In order to retrieve the chunk *brick-hat* it is sufficient to store only *brick* or only *hat* in STM. But what about the case of chunking a random sequence of binary digits by recoding them into octal? By definition, a random sequence contains no redundancy and should therefore not be compressible. The redundancy here follows from the fact that when coding a binary sequence verbally we use only two words (zero/oh/nought, one) out of the set of thousands of possible words in the language. For example, there are well over two-thousand

monosyllabic words in English, so each monosyllabic word could code approximately 11 bits. However, a word used to represent 0 or 1 codes only a single bit. In effect, the remaining 10 bits are redundant (predictable). In theory we could go well beyond encoding binary digits as octal and assign a separate monosyllabic word to each possible 11 bit binary number, although this might take quite some time to learn. With a memory span of six words it should then be possible to remember a sequence of 66 binary digits. The redundancy involved in coding binary numbers using two words is analogous to using each word in a 16-bit computer to code a single bit; 15 of the bits will be completely redundant. We will return to this point later: whether the capacity of a memory system appears to be limited by chunks or information will depend on the granularity of the representations available (e.g. bits, phonemes, words). The larger the units of storage, the more the system will appear to have a chunk-based limit.

Pointers. As suggested by Cowan and Chen, one way to implement compression by using chunking is to assume that items in STM are replaced by an index to the representation of the corresponding chunks in LTM (see also Huang & Awh, 2018) . However, treating STM as a set of pointers or indexes has implications that are rarely appreciated. The pointers themselves require memory capacity, which is a function of the amount of information that the pointers can encode. That capacity will be determined by the size of the set of elements that the pointers can address. The larger the set of elements that the pointers need to address, the more capacity will be required. Consider the case where we have a memory capacity of 16 bits. We could store the outcome of 16 coin tosses using 1-bit pointers, or a single pointer to one of 65536 words in a person's lexicon. In the

former case long-term memory would just contain the two entries ‘heads’ and ‘tails’. A single bit can point to or index the appropriate outcome. In the latter case, a pointer must have sufficient capacity to index any of the words in the lexicon. Note that it does not matter how long the words in the lexicon are, the function of the pointer is to select the appropriate word from the set of possible words. A pointer that could address any possible representation that might be stored in human LTM would have to contain a very large number of bits indeed.

One might suggest that pointers need only address representations in a subset of LTM, such as the set of chunks that have been encountered in a particular experiment, but the same constraints on address space still apply. In fact, the constraint is not simply on the size of pointers, but more generally on the amount of information required to code an item that occurs with a given probability. The number of bits required to code an item is given by its self-information:

$$I = -\log_2(p_i)$$

where p_i is the probability of item i. The more items there are in a given set, the lower will be their average probability and the greater their self-information, and the more bits will be required to encode them. Importantly, the number of bits required to code an item is not simply a function of the number of items, but also of the probability of each item. Items that occur infrequently are more surprising. They therefore contain more information and require more bits to code optimally. The message here is that memory capacity is a trade-off - for a given capacity we can store a lot of items from a small set or a smaller number items from a larger set. The limiting factor in STM is not the size of the chunks – storing the chunks

themselves is the task of LTM – but the size of the pointers needed to select the appropriate chunk.

Miller suggested that "Since memory span is a fixed number of chunks we can increase the number of bits of information that it contains simply by building larger and larger chunks, each chunk containing more information than before". However, increasing the size of the chunks does not necessarily increase the amount of information that can be stored. Imagine learning the words from the Bible and the works of Shakespeare as two very big chunks. LTM would need to contain all of the information necessary to accurately encode the sequence of words in each. However, a sequence of those two large chunks can be coded in STM by one bit per chunk, so long as the sequence never contains anything other than those specific chunks. Increasing the size of the chunks does not increase the size of the pointers required to encode them in STM or the amount of information that can be stored in STM.

Algorithms for data compression. One of the most familiar uses of data compression is in creating computer zip files. Zip files are an example of lossless data compression. When the files are uncompressed the original data can be recovered perfectly. In contrast, with lossy compression, some information is thrown away. The most familiar examples of lossy compression are MP3 encoding of audio signals and JPEG encoding of images. In both cases the original signal can only be approximately recovered from the compressed file. There is a trade-off between the degree of compression and the fidelity of the recovered signal. The greater the compression, the less accurately the signal can be recovered. It might seem that lossy compression must always be a bad thing and should be avoided unless there are strong limits on storage capacity. However, as we will see later, in the context of human

memory and perception, compression can have the benefit of revealing important information while discarding unwanted information.

One feature shared by many forms of data compression is that there are two parts to the compressed representation: the compressed code itself, and a codebook or dictionary (or possibly a program) that can be used to translate the compressed code back into its full form. For example, because the words 'chunk' and 'compressed' occur frequently in this manuscript we could compress the text by replacing all occurrences of those words with a shorter sequence of letters (perhaps 'CH' and 'CO'). However, in order to recover the original text we must also store the codebook (CH = chunk, CO = compression) along with the compressed sequence. We might consider the entries in the codebook to be analogous to the verbal labels for chunks proposed by Shiffrin and Nosofsky. The codebook enables the compressed representation (chunk labels) to be converted back into its original form. Note that there need be no systematic relationship between the form of the chunks in the original message and their codes. The code for 'compressed' could equally well be '97'. An efficient compression scheme can be thought of as one which minimizes the combined size of the code plus codebook. This is the principle behind Minimum Description Length (MDL) encoding (Grünwald, Myung, & Pitt, 2005; Rissanen, 1978), where length is measured in bits. For example, if a particular word occurred only once in a message, replacing that word with a code representing a chunk would actually increase the combined length because that word would still have to have an entry in the codebook. In the MDL framework, compression might also be achieved by creating a program that could reconstruct the input. The series of ascending real numbers would take an infinite amount of storage, but can trivially be encoded in a very short program.

MDL is best considered as providing a metric for evaluating the relative success of different compression methods. An ideal compression method would be one which minimized the redundancy in the input. MDL does not provide a specific algorithm for generating a compressed code. One of the simplest algorithms for performing compression is Huffman coding. This constructs an economical representation of its input by taking account of the relative frequency of symbols in the input such that more frequently occurring symbols are replaced with codes requiring fewer bits than less frequently occurring symbols. That is, shorter codes are assigned to items with lower self-information. An illustration of a Huffman code is given in Table 1.

==========

Insert Table 1 about here

==========

Note that Huffman coding does not perform any kind of chunking. The coding of any character is determined only by its relative frequency; characters will never be combined to form larger chunks with distinct representations.

The idea that STM might be able to benefit from data compression raises two intriguing questions: what are the codes, and where is the codebook stored? An important feature of the MDL framework is that it applies to the combined length of the code and codebook. In psychological accounts of chunking such of that of Cowan, the usual assumption is that the chunks themselves are stored in LTM and that that these are different from the representations in STM. That is, the compressed codes are in STM and

the codebook is in held LTM. For example, when recoding binary to octal, the mapping between binary and octal must already be stored in LTM. Incoming sets of three binary digits are replaced in STM by the corresponding octal digit and, at recall, those octal digits in STM must be used to access LTM and read out the corresponding binary digits. The chunks, or entries in the codebook, are constructed on the basis of experience of previously encountered material rather than being constructed afresh as each new set of items to be remembered comes along. It might seem that we could therefore expand the codebook without limit, as there are few practical constraints on the capacity of LTM. However, by analogy with pointers discussed earlier, the larger the codebook, the more information the codes need to contain to address the appropriate entry in the codebook. If the codebook is made larger than absolutely necessary, this will decrease the number of items that can be stored in STM.

Modality-specific stores. One implication of these considerations for human memory is that the best way to maximize the number of items of a particular kind that can be stored is to have a specialized store dedicated to one particular type of information (see Norris, 2017). More specifically, to have a store that represents only items from a clearly defined small set. Imagine that there is a cognitive process involved in understanding language that operates specifically on phonological information and never requires access to information from other domains. If that process had to access a general-purpose amodal memory system, it would need some way of manipulating pointers to a very large address space. On the other hand, if it needed only to access a store containing phonological information, it could perform its computations using pointers to a much smaller address space. There are fewer than 50 phonemes in English. They could therefore be coded by less

than six bits each. Such a store would seem to correspond precisely to the phonological store of Baddeley and Hitch's (1974) working memory model. This suggests that specialized memory subsystems are most likely to be found where they can support processing that can be performed using a restricted set of representations.

The representations supporting verbal STM are primarily phonological. (Baddeley & Hitch, 1974). This has implications for the kind of chunking process that might operate in studies of verbal STM. When compression is achieved using Huffman coding or an MDL procedure, this necessarily changes the form of the representations - that is, the representations would no longer be phonological. In these schemes the message is replaced by a compressed code expressed in a different vocabulary from the original message. Compression within a phonological store would therefore need to be achieved by replacing one phonologically expressed representation with another. This could be done in the way suggested by Cowan and colleagues, for example, by storing only the first word in a chunk. Alternatively, a chunk may be replaced with a completely different phonological form, as in the case of recoding binary as octal. Note that both of these schemes only work because the experimental procedures limit the number of potential messages that can be conveyed. Coding by storing the first word of a chunk will only be effective if there is only ever one chunk beginning with a given word, otherwise there would be no way of using that word to retrieve the correct chunk. Recoding binary to octal is only possible because the message is known to contain only binary digits. This link between compression and the set of possible messages corresponds to a central result in the field of information theory: efficient storage and transmission of information depends on knowledge of the statistical properties of the signal (Shannon & Weaver, 1949).

The question of what a compressed code might look like is far harder to answer. As already noted, in standard compression algorithms there need be no simple relation between the form of the code and the form of the chunk. In the case of recoding binary digits as octal, the compressed representation is in an entirely different vocabulary from the input.

Even the words in the language can be seen as a compressed code – a code that is used for communicating meaning. A simple way to characterize speech communication is as a way of transferring a sequence of words between two speakers. An efficient way to achieve this might be to use a compression scheme like a Huffman code where the most commonly occurring elements are represented by the shortest codes. Zipf's law (Zipf, 1935) indicates that this is indeed what happens - more frequent words tend to be shorter than rarer words. This enables information to be transmitted at a faster rate, and the number of words that can be stored will be greater than if word length was not related to frequency.

Compression and learning. So far we have presented the discussion as though the only point of data compression was to squeeze more information into a given amount of memory. However, data compression follows naturally from something that is even more important than making efficient use of memory: learning. Data compression is only possible by virtue of having learned something about the regularities in the data. As noted by Grünwald (2007) "the more we are able to compress the data, the more we have learned about the data" (p *xxv*). For example, the infinite sequence of ascending even numbers would be impossible to store in any memory system. But once one learns the underlying regularity it can be compressed into very short computer program. However, learning the

regularities is a difficult problem with no guarantees of success. Consider the following series of digits:

4159265358979323846264338327950288419 71..

This series can actually be represented very economically because it is the value of π starting from the third digit, but it is unlikely that anybody would discover that fact.

Data compression in the form of chunking plays a central role in several models of learning (Gobet et al., 2001; Orbán, Fiser, Aslin, & Lengyel, 2008; Robinet et al., 2011; Servan-Schreiber & Anderson, 1990). For example, Robinet et al. developed a model that uses Minimum Description Length (MDL) principles to construct a hierarchy of chunks. Their model successfully simulates a range of data on artificial grammar learning. Orbán et al. (2008) used a Bayesian approach to model visual category learning.

Compression and language. Although compression, in the form of chunking, may increase the amount of information that can be stored in STM, compression will also play a role in determining the nature of the linguistic representations underlying verbal STM. We noted above that words in a language can be seen as having evolved to produce an efficient code, but how might an infant discover that code? How might an infant learn the words of their language given that natural speech does not always contain clearly marked word boundaries? Speech is not a random sequence of phonemes - there are recurring patterns which generally correspond to words. This means that the signal can be compressed. Both de de Marken (1995) and Brent and Cartwright (1996) used an MDL approach to identify words in a transcription of continuous speech. The recurring patterns become the dictionary or codebook of a compressed code. In this procedure, words in the input are not identified

simply by 'trying to learn the words' but by trying to find an economical representation of the input. That economical representation is effectively a set of chunks that are a good approximation to the set of words in the language. Of course, word acquisition involves much more than compression. When discussing grouping we noted that grouping cues determine which chunks are learned. Similar principles apply to infant word learning. For example, Johnson and Jusczyk (2001) found that 7.5 month olds were more influenced by segmentation cures (stress and coarticulation) than by statistical patterns in the input.

More recent work on word acquisition has been generally been expressed in a Bayesian framework rather than MDL (e.g. Frank, Goldwater, Griffiths, & Tenenbaum, 2010; Goldwater, Griffiths, & Johnson, 2009). However, there is a close formal relationship between the two in that the central idea of both is to discover an economical (compressed) representation of the input.

These models of word acquisition operate on a phonemic transcription of the input. But a phonemic transcription is already highly compressed relative to the original acoustic input. The full processing stream of speech recognition involves moving from an informationally rich stream of acoustic data to a representation in terms of meaningful linguistic units such as phonemes or words. This entails a massive degree of data compression. Assume that we begin with a CD quality monaural signal with 16-bit samples at 44.1kHz. This corresponds to a data rate of about 700kbps (kilobits per second) and is more than enough to cover the full range of human hearing. The same signal can be compressed using MP3 encoding at 160kbps with negligible loss of fidelity to the human ear. MP3 coding at only 32kbps is still quite sufficient to transmit intelligible speech. Alternatively, a good speech codec can reduce this to about 800bps, and to transmit a

phonemic transcription would take only 50-100bps. This in turn can be reduced by at least a factor of 2 by taking account of the redundancies in human language. We can therefore compress the acoustic signal by a factor of over 10000 and still recover the original message. However, to achieve this degree of compression requires a huge amount of knowledge. Even the process of constructing an MP3 encoding requires knowledge about human hearing and the importance of critical bands. The acoustic waveform reconstructed from a high quality MP3 might not be high quality to an organism with a different hearing system. Converting an acoustic waveform into a phonemic transcription requires a deep understanding of the regularities of human speech. Even the best automatic speech recognition systems can only approximate this level of compression under ideal conditions - they have not yet learned enough about the regularities of human speech.

The task of recognizing words from a continuous acoustic input therefore involves a huge degree of data compression. However, the goal is not compression, but understanding. Converting the acoustic signal to some form of linguistic representation gives us data compression for free. We now have a signal that we can hold in a store with quite limited capacity. However, there is one very important consequence of such extreme data compression; all we can place in that store is speech, because the resulting code is incapable of representing anything else. The difference in timbre between a note played on a piano and a trumpet, for example, simply cannot be represented in terms of phonemes. Even a richer speech representation that also included, for example, prosody, would also fail to encode this distinction. This is another manifestation of the trade-off discussed in the context of pointers. We can use a given amount of memory to store a large number of things from a small set (phonemes) or a small number of things from a large set (arbitrary

sound waves). It also demonstrates that this is a form of lossy data compression – information is thrown away that cannot be recovered. But most of the information that is lost plays no role in understanding speech. More specifically, the information that is thrown away plays no role in the understanding of the speech of our *native* language. Short-term memory for non-native speech is much poorer than for speech in our native language (Thorn & Gathercole, 1999). This, for example, offers a possible explanation of why nonwords words with a high phonotactic probability are easier to repeat than those with a lower phonotactic probability (Gathercole, Frankish, Pickering, & Peaker, 1999); they can be coded more economically in phonological STM.

The fact that there are constraints on how compression might be achieved in a phonological store should not be taken to imply that the representations are completely fixed. The nature of the phonological or phonetic code must be capable of changing with experience in order to acquire one's native language, or to learn a new language. Different languages have different phonological repertoires, and language acquisition therefore depends on tuning the store to support the necessary phonological representations

The ability to operate on a compressed representation of an acoustic signal is not something unique to humans. Neurophysiological recordings from the primary auditory afferents of frogs shows how the auditory system is tuned to the statistics of the environment. Rieke, Bodnar, and Bialek (1995) found that the rate of information transmission of spike trains was 2-6 times higher for naturalistic sounds than for broad-band stimuli, and that this rate was close to the theoretical maximum.

In contrast to our view, Jones and Macken (2018) (see also: G. Jones, 2016; G. Jones et al., 2014; G. Jones & Macken, 2015) have suggested that phenomena such as the

phonotactic probability effect have nothing to do with a dedicated phonological store. They argue that "performance in vSTM measures primarily reflects domain-general associative learning" (216) and they assume that experience with linguistic input makes it possible to represent the input in terms of successively larger chunks. In line with Miller, they suggest that sequences comprised of fewer chunks are easier to remember. However, contrary to Miller, they wish to deny the existence of STM altogether - "our computational model of associative learning provides a parsimonious explanation of performance in vSTM tasks without the need for additional bespoke processes such as a short-term memory system" (p226). They used their model to explain the effects of phonotactic probability on nonword repetition (G. Jones et al., 2014), and have suggested that it explains why digit span is better than word span (G. Jones & Macken, 2018) . In the simulations they report, their measure of performance is simply the number of chunks that the model requires in order to represent an input list (or a nonsense word). The model has no mechanism for encoding, storage, or recall of those chunks. What's missing from this model is somewhere to store the sequence of chunks that the input is recoded into: a short-term memory system.

Nevertheless, their data make a good case that input sequences that can potentially be encoded in terms of fewer chunks are easier to remember. Such sequences will be easier to compress. The unresolved question though, is - where is the compressed code stored? In the case of the very local statistical dependencies between phonemes that might underlie nonword repetition effects, we suspect that the code and codebook may be part of STM itself. For larger-scale dependencies such as those present in digit sequences, it is much less clear. They might be better explained in terms of the redintegration mechanism to be described later.

The chunking process suggested by Jones and colleagues has parallels with a much earlier algorithm for redundancy reduction developed by Redlich (1993). This was not intended as a psychological model. Both procedures operate by combining existing chunks into larger chunks. However, in Redlich's procedure, larger chunks (Redlich refers to them as features) are constructed only when they reduce the redundancy in the representation of the input corpus. The Jones model simply creates ever larger chunks regardless of whether they can help compress the corpus.

Compression should never be seen as an end in itself. For example, both MP3 and FLAC encoding of auditory signals operate on a purely acoustic level of representation that does not respect the linguistic structure of the input. Both may offer a high degree of compression of a speech signal, but the compressed representations will obscure the properties of the signal that are crucial for speech recognition. That is, although the acoustic signal can be reconstructed (perfectly in the case of FLAC, or approximately in the case of MP3), the compressed code itself obscures critical information. This is a point made very clearly by Barlow (2001): "coding should convert hidden redundancy into a manifest, explicit, immediately recognizable form, rather than reduce it or eliminate it" (p246). In the case of speech, that recognizable form might take the form of, for example, phonetic features or phonemes.

Evidence for compression. Although chunking can be seen as a form of data compression, studies of STM have rarely considered chunking in this context (although see Brady et al., 2009; Chekaf et al., 2016; Mathy & Feldman, 2012). Mathy and Feldman proposed that serial recall from STM involves data compression. They found that participants had better memory for lists of digits when they contained runs of increasing or

decreasing digits. They suggested that the presence of regular patterns enabled the lists to be compressed so as to take up less capacity. What was lacking from this account was any suggestion as to the form that the compressed representation might take, or where the codebook might be stored. If participants can compress the run of digits 4, 5, 6, 7, then not only must STM contain a more economical representation of this sequence, but it must also have access to a codebook (in LTM) to permit the code to be converted back into the sequence of digits. One possibility might be to incorporate some representation of ellipsis such as '4-7' where '-' codes the sequence of digits in between the first and last. "5, 6, 7, 8" could be represented as '5-8'. Here '-' would represent a different set of digits from "4, 5, 6, 7" because it would encode "6, 7" instead of "5,6". We noted earlier that in the MDL framework, compression can be achieved by being able to generate the input from a program. The compressed code for '4-7' might indicate that the first and last digits should be passed to a program to generate the appropriate run of intervening digits. Of course, to achieve compression the call (pointer) to the program must be coded more economically than the sequence of digits it replaces.

In a later paper Chekaf et al (2016) examined recall of sequences of symbols constructed from combinations of the three features: large/small, square/triangle, black/white. The critical comparison was between a "Rule" condition where the symbols systematically cycled through the features (e.g. large white square, large black square, small white square, small black square, small white triangle, small black triangle) and a "Dissimilar" condition where there was no systematic relationship in the order of the symbols (e.g. large white square, small black triangle, large black square, small white triangle, small black square, small white square). Recall was better in the Rule condition,

even though all of the sequences were novel. This extends Mathy and Feldman's (2012) result by showing that people are able to take advantage of regular patterns in sequences even when those patterns do not correspond to pre-existing chunks. Chekaf et al. suggested that people were able to take advantage of the regularities to perform data compression. They claimed that this could not have been achieved by an MDL method because the sequences to not contain recurring patterns. However, as already noted, MDL coding does not need to be achieved via a codebook and can, for example, recode the input as a program. Recoding using a set of rules is perfectly consistent with the MDL framework. For example, by adding appropriate syntactic symbols combined with appropriate semantics in LTM, the Rule sequence above might be coded more economically as [white, black] (large square, small square, small triangle). Note that the code needs to contain enough information to distinguish unambiguously between different possible rule-based sequences.

One of the strongest indications of a genuine effect of data compression comes from a study of visual STM by Brady et al. (2009). They performed two visual STM experiments where participants had to report the color of one of eight rings or circles that had been presented for 1s. In their first experiment, the display consisted of four circles arranged in a diamond. Each circle had two concentric rings of different colors, and the displays always contained eight different colors. After the display disappeared, participants saw a second display that indicated which ring they had to recall. Brady et al. varied the frequency with which different colors were paired together in the circles. Individual colors always appeared with equal frequency. Not surprisingly, colors appearing in the pairs that were presented more frequently were recalled better. Furthermore, their estimate of the number of items that could be held in memory increased over the course of the experiment. However, the

critical finding was that, in displays containing high probability pairs, recall of colors in low probability pairs also improved relative to displays containing only low probability pairs. This implies that the presence of high probability pairs allows the representation of the entire display to be encoded more efficiently. The data can be interpreted in terms of chunking on the assumption that high probability pairs come to be treated as chunks which take up less memory capacity and free up space for pairs that do not form chunks; chunking leads to data compression.

Brady et al. constructed a Bayesian model of how their participants learned the probabilities of pairs. Treating each pair as a separate item (i.e. blue outside, red inside might be pair *x*) they showed that performance was inversely correlated with the length of the Huffman code for the display. Even though memory capacity as measured in terms of number of items stored increased over the course of the experiment, the estimated capacity expressed in bits, as derived from the effective Huffman code, remained constant. That is, there was no evidence that practice with the task improved underlying memory capacity, but it does improve how that capacity can be used. Although Brady et al. determined memory capacity in terms of the length of a Huffman code, they are neutral with respect to whether compression is performed by something akin to Huffman coding, or by chunking. Indeed, they show that a chunking model can also give a good account of their data. They suggest that chunking can be thought of as a discrete approximation to a more graded form of compression. However, as they note, the standard Huffman coding algorithm is probably a poor model of the psychological processes involved. Huffman coding is ill-suited to

modelling the process of adapting to changes in probabilities over time[1]. A Huffman code would need to be continually recomputed over all possible color pairs and the codes assigned to different pairs will change as their probabilities change. In effect, the label assigned to a particular chunk would keep on changing throughout the course of the experiment. If some items come to be presented more often, this naturally changes the relative probabilities of all items and the codes for all items will need to be updated.

Brady et al's data provide an important qualification to Miller's claim that "The span of immediate memory seems to be almost independent of the number of bits per chunk" (p93). In Brady et al's experiments there is an improvement in performance due to chunking, but the memory capacity in bits remains constant. Chunking has enabled that capacity to be used more efficiently in coding the choices that must be made in the experiment. In line with the arguments being presented here, Brady et al suggested that information theory could provide useful constraints on theories of chunking. Brady et al's findings are specific to visual STM. We will see later that there are parallel findings for verbal STM (Norris & Kalm, 2018).

Chunks or information? Having discussed data compression and the possible role of learning in developing compressed representations, we can now reconsider Miller's arguments against the idea that capacity might be determined by the amount of information that can be stored. Is there really a categorical difference between a chunk-based and an information-based limitation on STM capacity? Miller based his conclusions on

[1] The rather more complicated Adaptive Huffman code can update the code as it processes a message, but is less efficient.

the studies by Hayes (1952) and Pollack (1953) which had found that span varied little as a function of the amount of information per item.

However, span could only be a function of information if participants have developed an optimal compressed representation of the stimuli prior to the testing phase of the experiment. Furthermore, this representation must be different for each category of stimuli whose storage is to be optimized. An optimal compressed code that was developed for the set of digits used in an experiment would be unable to code anything other than digits. If, having been tested on digits, participants were tested again on letters of the alphabet that would require a completely different code. But, outside of the laboratory, digits or letters are words like any others. If participants have already developed an efficient code for words in general, they will have to replace that code with one specific to digits or letters in order to make full use of the information capacity available. The important message here is that even if memory could hold only a fixed number of bits, capacity could only fully utilized if compression is optimal, and that means tailored to a specific set of stimuli.

A further factor that will prevent memory capacity being a simple function of information is the granularity of the representations that STM can support. As noted earlier, the larger are the units of representation, the more the system will appear to have a chunk-based limit. If the smallest unit of representation is a bit, and the compression is optimal, capacity will be determined by the number of bits available. However, if the smallest units of representation are, say, 16-bit patterns, perhaps corresponding to each of the words in an individual's lexicon, this will impose strong constraints on the sort of compression that is possible. For example, it will be impossible to use Huffman coding to recode individual

words or digits into a more economical representation because the Huffman code operates at the level of the bit. As an illustration, we used Huffman coding to compress a seventy-thousand-word dictionary. With no compression, each word would take 16.1 bits. With compression, a handful of very common words have codes of four bits, and the most infrequent words have codes of 21 bits. If each unit of representation in memory was allocated a fixed number of bits, the 'compressed' representation would actually require more capacity to store (a minimum of 21 bits per word) than the uncompressed representation (16.1 bits per word). Compression will therefore depend on how efficiently the input can be recoded into the available representational units.

Rather than thinking of capacity as being determined either by information or by chunks, it is perhaps more useful to think of it as being a continuous function of the density with which information can be packed into memory. With small units of information and optimal recoding, capacity will appear to be determined by information. With larger units, especially when recoding options are limited (e.g. to words) capacity will appear to be determined by chunks. In other words, the question is not whether capacity is determined by information or chunks, but how efficiently the input can be compressed.

The discussion of compression demonstrates that information-theoretic considerations place a number of constraints on how chunking might operate. First, there are fundamental limitations in the amount of information that can be coded by chunks. As the number of possible chunks that might need to be stored in STM increases, so will the number of bits required to code an index or pointer to a chunk. Ideally, representations in STM (codes or pointers) should be optimized to store information efficiently (that is, description length should be minimized). Second, the potential for compression will depend

of the flexibility and granularity of the representations that STM can support. An inflexible system which has been fine-tuned to represent one kind of information economically may not be able to adapt to perform optimal compression of other material. This will apply even in the case of trying to compress digits in a vocabulary developed to represent words. There will be no problem representing the digits, as they are just a subset of the words; the problem will be in developing an alternative and more compressed representation.

==========

Insert Box 1 about here

==========

Redintegration and Bayesian inference

The idea that chunking may be a form of data compression conforms to the informal notion that chunks help us squeeze more information into a given amount of memory. An alternative view of chunking is that representations of chunks exist only in LTM, but can be used to reconstruct a degraded trace in STM by a process often referred to as redintegration (Brown & Hulme, 1995; Hulme, Maughan, & Brown, 1991; Hulme et al., 1997; T. Jones & Farrell, 2018; Lewandowsky & Farrell, 2000; Poirier & Saint-Aubin, 1996; Roodenrys & Miller, 2008; Schweickert, 1993; Stuart & Hulme, 2009; Thorn, Gathercole, & Frankish, 2002). Redintegration offers a simple explanation of the fact that, for example, words are easier to remember than nonwords (Hulme et al., 1991; Hulme, Roodenrys, Brown, &

Mercer, 1995) or that STM for high-frequency words is better than for low-frequency words (Hulme et al., 1997), but can apply equally well to chunks at other levels.

According to this view, the contents of STM are not recoded into chunks on input. Chunks in LTM come into play only at recall and do not alter the contents of STM. If information in STM is degraded as a consequence of forgetting, items or chunks with pre-existing LTM representations can be recovered on the basis of less information than would be required to recover items with no LTM representations.

Redintegration can be achieved by treating recall from memory as a process of Bayesian inference whereby representations of chunks in LTM provide the priors that can be used to interpret a degraded representation in STM (Botvinick, 2005; Botvinick & Bylsma, 2005). According to this view, STM would be a passive process whose contents remain the same regardless of whether they correspond to chunks or not. Treating memory retrieval as a process of Bayesian inference has a long history in the study of memory (Botvinick, 2005; Botvinick & Bylsma, 2005; Crawford, Huttenlocher, & Engebretson, 2000; Hemmer & Steyvers, 2009a, 2009b; Huttenlocher, Hedges, & Duncan, 1991; Huttenlocher, Hedges, & Vevea, 2000; Shiffrin & Steyvers, 1997; Sims, Jacobs, & Knill, 2012; Xu & Griffiths, 2010). In effect, chunks in LTM are hypotheses with associated priors, and recall involves computing the posterior probability of those chunks given the evidence in STM. Thus, the identity of more probable representations (those corresponding to chunks in LTM) can be inferred from STM on the basis of less evidence than would be required for less predictable

representations[2]. For example, the letters FBI form a familiar chunk with a higher prior probability of occurring together than a random sequence of letters such as SVY. Given an equivalent amount of evidence from a degraded representation in STM the letters FBI will have a higher posterior probability than the letters SVY. The fact that FBI forms a chunk will therefore benefit recall.

==========

Insert figure 1 about here

==========

Figure 1 provides a schematic comparison of the processing stages involved in chunking, redintegration and, for comparison, an ideal observer. We assume that linguistic encoding transforms the sensory input into a representation such as phonemes or words. The sensory input might be either spoken or written. In the case of chunking, the LTM recoding process performs a mapping between the linguistic representations and chunks e.g. binary digits to octal digits. The chunks are represented in the same linguistic vocabulary, and it is these representations that are stored in STM. At retrieval, LTM is used to decode the chunks back into the linguistic units.

[2] The computations are identical to those used in models of word recognition such as Norris (2006) and Norris and McQueen (2008) where high-frequency words can be identified on the basis of less evidence than low-frequency words. The sole difference is whether the evidence comes directly from perceptual input or from representations held in STM.

In the case of redintegration there is only linguistic encoding. LTM comes into play only at the decoding process. Both chunking and redintegration will lead to improved memory for inputs containing chunks. We assume that linguistic encoding is optimized to code spoken language efficiently, but not flexible enough to dynamically adapt to the statistical properties of the input on a trial-by-trial basis. For example, even if the input on a particular trial is the same word repeated several times, it cannot take advantage of this redundancy and generate a more efficient representation that can be stored more economically in STM.

Note that the LTM decoding process in the chunking and redintegration schemes are different. In a slot-based model such as that proposed by Cowan, the LTM decoding might simply consist of recognizing that an item in a slot is some form of index or pointer to a chunk (e.g. ‘F’ or a pointer, representing the chunk FBI) and then replacing that index with the full chunk to generate a response (‘F’ -> ‘F’, ‘B’, ‘I’). In the case of redintegration the output from STM is not an index to a chunk but a degraded form of the input chunk itself (‘F’, ‘B’, and ‘I’, not ‘F’, and not an index or pointer to ‘FBI’) and LTM is used to reconstruct that degraded representation and produce a more reliable response.

The ideal observer has the same form as the ideal observer model of visual STM presented by Sims et al. (2012). In an ideal observer model the encoder would take full advantage of the statistical properties of the input to produce an efficient code that can maximize the amount of information that can be transmitted through STM. It is not limited to use any particular form of encoding. However, it should take into account the same linguistic regularities as the linguistic encoder. If all of these conditions are satisfied, memory capacity would be purely a function of information, as measured in bits.

Assuming that he capacity of STM in bits is the same in all three cases, only the ideal observer will appear to have a capacity limit determined by the amount of information that can be stored. If the input can be chunked, the chunking and redintegration models will appear to have a capacity determined by a fixed number of chunks. If not, capacity will appear to be determined by the number of items that can be stored. The difference between the models is in the nature of the representations generated by the encoding process. Linguistic encoding imposes a granularity on the representations that can be stored in STM.

Evidence for redintegration. The case for redintegration has most often been made in the context of the recall of individual items. For example, Hulme et al. (1997) found that low-frequency words were sometimes erroneously recalled as a similar sounding high-frequency word ('list' substituted for' lisp'). However, redintegration can alo operate over larger units or whole lists. More direct evidence for a Bayesian account of recall from STM comes from a study by Botvinick and Bylsma (2005). They trained participants to recall sequences of nonwords generated by an artificial grammar and found that their performance improved over time as a function of how likely the sequences were to be generated by the grammar. A similar result has been reported by Majerus, Van der Linden, Mulder, Meulemans, and Peters (2004). In some respects these results are similar to Mathey and Feldman's finding that sequences of digits are recalled better when they contain runs of ascending or descending digits. The main difference is in terms of whether the familiarity of the sequences comes from pre-existing knowledge or from experimental training. A critical finding in Botvinick and Bylsma's's study was that, for sequences of equal probability, recall was worse if neighboring sequences were also of high probability. They

referred to this as the ‘good neighbor’ effect. This is exactly what would be expected from a Bayesian view. The basic assumption of the Bayesian framework is that participants will recall the sequence with the highest posterior probability. That will be a function of both the evidence for each possible sequence (the likelihood) and the prior probability of those sequences (determined from experience with the grammar). Recall will therefore be biased towards higher probability sequences and this will reduce the probability of correctly recalling a sequence with a high probability neighbor. This kind of result is difficult to reconcile with many data compression schemes. In a Huffman code, for example, there is no relationship between the form of the original symbols and the form of the compressed code. The same is true of many MDL codes. Although the codebook must contain the mapping between the input symbols and the code, the form of the code itself can be completely orthogonal to the form of the input. More specifically, sequences that are neighbors in the original message space need not be neighbors in the compressed representation. If that is so, a compressed code for a sequence would show no systematic neighborhood effects.

Whereas Botvinick and Bylsma used an artificial grammar, Jones and Farrell (2018) studied the influence of English syntax on memory. They found that recall of syntactically ill-formed sequences tended to be biased towards producing sequences that conformed more closely to English syntax. Like Botnivick and Bylsma they argued that this was best explained in Bayesian terms as a bias on recall of order. Interestingly, in these experiments the bias could not operate at the level of the order of individual words as most of the words would never have co-occurred before. The bias must operate at the level of syntax. Jones and Farrell compared four different models, all implemented within Henson's (1998) Start-End

Model. However, although they refer to one of these models as a chunking model, syntactic chunks simply determined the source of the priors rather than the way the words were coded in memory. Nevertheless, they concluded that their results were more consistent with redintegration than chunking.

Although compression cannot easily explain the data from Botvinick and Bylsma or from Jones and Farrell, redintegration has no explanation for the recorded cases of exceptional digit recall. Individuals with a very large digit recall capacity invariably adopt complex recoding or chunking strategies during presentation (e.g. Ericcson et al., 1980; Ericsson, Delaney, Weaver, & Mahadevan, 2004). In these extreme cases it is unlikely that redintegration has any significant role to play in memory. If it did, this would imply that someone like SF could store a partially degraded sequence of 79 digits in STM and then use his knowledge of running times to reconstruct that degraded trace. However, in conventional STM studies with untrained participants, both data compression and redintegration might operate.

Is there compression in verbal STM? The clearest evidence of compression comes from Brady et al's study of visual STM where performance on low probability items was improved by the presence of high probability items in the same display. However, it is not at all clear whether one might expect to see similar signature of compression in verbal STM, especially if the representations supporting verbal STM are primarily phonological. Consider an experiment similar to those of Cowan and colleagues where chunks are induced experimentally by exposing participants to specific pairings of words, and where all individual words are seen equally often – all that differs between chunks and other words is

the pairing of the words (their mutual information). The chunks introduce redundancy into the input and there is therefore the potential for compression. If the input is treated as a sequence of words, both unchunked single words and chunked pairs contain the same information and should be represented by codes of the same length. But what if the input is represented purely phonologically? Changing the probabilities of sequences of words (their mutual information) will not necessarily have a significant impact on the statistical regularities at the phonological level. The redundancy introduced by the presence of word chunks will be much less apparent and will only emerge if the system is sensitive to statistical relationships in the input spanning many phonemes. The difference can be illustrated by considering Shannon's (1949) examples of a second-order approximation to English letters and a second-order approximation to English words:

Letters: ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Words: THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

Both compute statistics over the same number of units but the second-order letter approximation contains no useable information about the associations between words. A store that is sensitive only to phonological information is therefore unlikely to adapt its coding scheme to regularities that are only present at a lexical level. This places strong constraints on the kind of compression that is possible. Compression will only be possible by changing the mapping between phonological codes and chunks. This restricts us to chunking by verbal *recoding* or *deletion.*

However, this shouldn't be taken to imply that there can be no compression in a store that is phonological as this is what must happen when learning a language. Compression in a phonological store might be more likely in the incidental phonotactic leaning paradigm used by Majerus et al. (2004) where the associations were between successive phonemes and successive CV syllables.

Distinguishing Between Compression and Redintegration

Given that both data compression and redintegration predict that lists of items containing familiar chunks will be easier to remember than those that do not, is it possible to distinguish between them? Fortunately, there are some conditions where they make different predictions. Consider the task of recalling a list where only some of the items form chunks. The compression view would imply that the formation of chunks would release capacity which could be used to store more items. Consider a slot model where each chunk occupies a single slot. If chunking frees up an extra slot in memory, then that slot can be used to store one more single item or one more chunk. That is, chunking should benefit all items in the list. Furthermore, if items can be combined to form chunks, each of those chunks takes up only a single slot. Therefore, whenever items are combined into a chunk this will release at least one extra slot that can be used to store and additional item or chunk. Consequently, any benefit of decreasing the number of chunks in a list (i.e. increasing the size of the chunks) should be associated with improved recall of all items in the list, not just the multi-word chunks themselves. This would be analogous to what Brady et al (2009) found with chunking in visual STM. They found that, on trials in which there were more high-frequency color pairs, memory was also better for the lower frequency pairs. This is,

when there were high frequency chunks in the display, this appeared to free up capacity which could be devoted to lower frequency pairs too.

A further prediction of a simple slot model is that the presence of chunks should benefit multi-word chunks and singletons to the same extent. The probability of storing each chunk is the same as the probability of storing each singleton; chunks and singletons have the same probability of occupying one of the slots. Given that the probability of remembering an item in multi-word chunk must be the same as the probability of remembering the chunk, the probability of recalling any single word will be the same regardless of whether or not it is part of a chunk. The chunking model therefore predicts that chunking will improve the recall of all items in the list, and they will all be improved to the same extent.

In contrast, the redintegration view predicts that chunking should have no influence on the amount of information or the form of the representations actually stored in STM. Chunking does not influence the contents of STM, merely how well they can be retrieved. Consequently, redintegration will benefit only those items that form chunks. Overall performance will improve as the number of chunks in a list increases, but this should be entirely due to superior recall of the chunks. The performance on the remaining items should remain constant.

Norris and Kalm (2018) tested these predictions in a series of experiments using similar methods to those of Cowan and colleagues. In their first three experiments they familiarized participants with two-word chunks. Experimental lists were seven items long and could contain zero, one, two, or three, two-item chunks. The remaining items were equally familiar singletons that had not been learned as part of a chunk. Memory was tested

by means of either 2AFC recognition (experiments 1 and 2) or serial recall (experiment 3). As would be expected, in all experiments recall improved as a function of the number of pairs in the list. Using 2AFC, performance on the pairs and the singletons themselves was completely unaffected by the number of pairs in the list. This is the result expected if the benefit of chunking were attributable to redintegration. Increasing the number of pairs did not seem to free up capacity that could be used to improve memory for other items. The overall improvement on lists containing more pairs was entirely due to the fact that the individual pairs were recalled better than the singles. Therefore, the more pairs there are in the list, the better the list will be recalled. In serial recall the results were similar. Performance on pairs was better than for singles and did not improve as a function of the number of pairs in the list. However, performance on singles did improve with more pairs. Norris and Kalm suggested that this could be due to the fact that as the number of pairs in the list increases this places more constraint on where the singletons can appear.

Based on the idea that the process of encoding items as chunks and then decoding the chunks back into single items might have a cost, Norris and Kalm investigated whether people might be more likely to adopt a compression-based chunking procedure with larger chunks. With larger chunks, the benefits of active chunking might be more likely to outweigh the costs. Assume for a moment that people use a deletion code. Each chunk has to be identified as such, to be replace by the first word of the chunk and, at recall, the first word of the chunk must be used to retrieve the second word in the chunk. To benefit from active chunking all of this extra processing has to be balanced against the cost of remembering one extra item per chunk.

In three further experiments they therefore used pre-learned triples. They found that when lists contained more triples memory was better for both triples and singletons. This held for both 2AFC and serial recall. This study therefore found evidence for redintegration, with small chunks (pairs), and compression, with larger chunks (triples).

A further interesting finding from this study was that pairs and triples were always remembered better than singletons. This is not what would be expected from a pure compression view. Singletons and pairs or triples would each be stored as a single chunk, and the contents of the entire chunk should be either remembered or forgotten together. If the unit of storage is a chunk, there is no reason why larger chunks should be less likely to be forgotten. However, this result is not at all surprising if redintegration always plays a role in recall. If redintegration is beneficial, and it can operate with two-item chunks, there is no reason why it should not continue to operate with larger chunks and hence confer an advantage on triples over singletons. This indicates that redintegration is always at work in memory, but that active chunking only comes into play when its benefit outweighs the cost of encoding and decoding the chunks.

The cost of chunking. The cost of chunking is clearly illustrated by studies by Glanzer and Fleishman (1967), Pollack and Johnson (1965), Huang and Awh (2018) and Kleinberg and Kaufman (1971). Glanzer and Fleishman had subjects recall strings of nine binary digits presented simultaneously for 0.5s. Some of their participants were trained over nine days to recode binary digits as octal, and others to use different strategies, including ones of their own choice. After nine days of training one might have expected the binary to octal conversion process to become automatic. Based on the findings of Smith (cited in Miller,

1956) one might expect those trained to use the octal recoding strategy to perform best. In fact, they performed worst. Glanzer and Fleishman suggested that participants may not have been able to apply the strategy efficiently enough to benefit from it. Perhaps this isn't too surprising given that their participants would only have had 167ms to perform each of the three binary to octal conversions required to recode a nine-digit binary number.

Pollack and Johnson's task was less demanding but still revealed limits on how efficiently subjects could employ recoding strategies. They trained participants for twenty-eight 1.5 hour sessions to recode groups of four binary numbers as the decimal numbers 0-15. They varied the rate of presentation and found that performance dropped from near perfect at a rate of 0.7 digits per second to only 60% of lists being correct at a rate of 2.8 digits per second. Both of these studies highlight the fact that the benefit of chunking has to be weighed against the cost of recoding material into chunks in the first place.

Kleinberg and Kaufman tested memory for visual patterns constructed from displays of 13 illuminated dots in a 5 x 5 matrix. They found that memory was constant in terms of amount of information at fast presentation rates (less than one second) but constant in chunks for slower rates. Effective use of chunking required time.

Huang and Awh used a probe paradigm similar to that of Brady et al. (2009), where the stimuli could either be four color pairs or four letter pairs. The pairs were either familiar or not. In the case of letters the familiar pairs formed words. They found a benefit of chunking, but this disappeared when participants had to respond under time pressure. This study indicates that there is a cost in decoding chunks as well as in encoding them. In the chunking model favored by Cowan et al, the use of chunks is assumed to be probabilistic rather than all or none. We suggest that the cost of chunking may be one factor that

modulates the probability of forming and using chunks. The overall picture that emerges from these studies of chunking and compression in STM is therefore that there is evidence for both, but chunking by recoding more is costly than redintegration. Therefore chunking happens some of the time but redintegration happens all of the time.

Chunking other domains.

Much of what we have said about chunking in verbal STM also applies to other domains. De Groot's (1946) famous study of chess masters' memory for the arrangement of chess pieces first brought the concept of chunking into memory research. Similar results have been found for differences between experts and novices in solving physics problems (Larkin, McDermott, Simon, & Simon, 1980) and expert and novice programmers in remembering computer code (Ye & Salvendy, 1994).

Chunking has also been invoked as an explanation for better memory for regular than irregular patterns in visual short-term memory (Bor, Duncan, Wiseman, & Owen, 2003; Brady et al., 2009). Some of these results can be attributed to recoding into a different representational vocabulary. For example, configurations of chess pieces could be recoded as 'knight fork' or 'Opera mate'. What is common to all of these examples is the availability of an alternative code in LTM that can map between the visual scene and higher order labels. Furthermore, the input itself can readily be translated into a symbolic code. In chess, what needs to be remembered is the location and identity of each piece in the representational vocabulary of chess. There is no need to remember the orientation of the pieces, or their exact shape. What needs to be remembered is the standard label for the pieces (knight, rook etc.) and the square on the board they occupy. In contrast, in many

visual STM tasks it is the lower level perceptual details that have to be remembered; the exact physical location, orientation, color or configuration. For some arbitrary visual pattern there is no guarantee that this can be mapped onto a pre-existing representation in LTM. However, the representation of the input can potentially be compressed. To take a very simple example, a red square, a red triangle and a red circle can be more effectively compressed than a red square, a blue triangle and a green circle. In the former case, only one color has to be specified. Note that these principles apply regardless of whether statistical regularities can be coded as simple verbal expressions. Orhan and Jacobs (2013) have provided a formal account of how visual STM can take advantage of the statistical properties of a visual scene to encode it more economically and to improve memory performance (for review see Orhan, Sims, Jacobs, & Knill, 2014).

Discussion

According to Miller, chunking involves active recoding. In the simplest case this might involve nothing more than deleting redundant information. For example, participants might remember only the first word of a multi-word chunk. Alternatively, recoding may involve replacing one representation in memory with a completely different one, as when recoding binary as octal. A very different view is that chunks exist only in LTM, and the contents of STM are not recoded. The benefit of having chunks comes about from their ability to support redintegration of degraded traces.

In common with Cowan and others we suggested that the simplest way to implement chunking as data compression would be to consider chunks to be functionally equivalent to pointers to representations elsewhere in memory. This forces us to consider

the amount of information that might be encoded in a pointer. Pointers themselves must necessarily take up memory capacity. Those that have the potential to address a large number of chunks will take up more capacity than pointers to a small set of chunks. A given amount of memory can therefore address a large number of items selected from a small set or a small number of items selected from a larger set. Chunking can be seen as a means of using the pointers in an efficient way. For example, a pointer that could address any word in one's vocabulary would be used inefficiently if it only had to distinguish between the words *one* and *zero.* A pointer that could address the names of the digits 0-7 would make much better use of the available capacity. Recoding binary digits into octal will therefore improve recall.

Most demonstrations of exceptional memory performance have been achieved using relatively slow rates of presentation. For example, SF (Ericcson et al., 1980) received one digit per second. He explicitly recoded digit sequences into running times. For most of us, very few sequences of digits, other than consecutive runs of increasing or decreasing digits, have any great familiarity. We are generally poor at noticing even familiar sequences when they are embedded in longer sequences (75106661, 18191457, 3117767, all contain a significant historical date beginning at position 3). Perhaps individuals such as SF have a much greater repository of familiar numbers (chunks) in LTM, and become practiced in parsing the input into those chunks. In their study of the mnemonist Rajan, Ericsson et al. (2004) attributed his exceptional performance to extensive practice in memorizing numbers "such as phone numbers, dates and cricket scores and statistics", rather than to any increase in underlying memory capacity

Chunking requires an active recoding process and, as we have seen, comes at a cost. In contrast, when using redintegration, knowledge of chunks and statistical regularities in the input need only come into play at recall. Nevertheless, it may take extensive experience to internalize the necessary statistics. Botvinick and Bylsma (2005), who showed that serial recall was influenced by the constraints of an artificial grammar, had their participants practice for one hour a day over 15 consecutive days. Botvinick (2005) also used 15 sessions of training. On a shorter time-scale, Majerus et al. (2004) showed that participants' memory for nonwords improved following about 30 minutes exposure to syllables generated by a simple phonotactic grammar (3000 CV syllables).

Mathy and Feldman (2012) gave their participants only 100 lists and were able to show that their performance improved when lists contained runs of consecutive increasing or decreasing digits. However, in contrast to the more subtle manipulation used by Botvinick and Bylsma (2005), or the experimentally induced chunks used by Cowan and colleagues, using runs of digits may tap into well-established representations. They may even be susceptible to high-level cognitive strategies analogous to the binary to octal recoding studied by Smith (cited in Miller, 1956).

Compression and resources

The accounts of chunking we have considered so far all assume that memory capacity is limited by the available number of discrete representations such as bits, items, or chunks. A fundamentally different view of memory is that capacity is determined by a limited resource (Alvarez & Cavanagh, 2004; Bays, Catalao, & Husain, 2009; Fougnie, Cormiea, Kanabar, & Alvarez, 2016; Just & Carpenter, 1992) which can be flexibly distributed

across items in memory. As the number of items in memory increases, a common pool of resources will have to be distributed over a larger number of items.

Perhaps the simplest way to view resources is as the capacity of the store measured in bits. That capacity can potentially be distributed over the items in memory to vary the precision with which they are coded[3]. In the brain this could perhaps be implemented in terms of a resource limited pool of spiking neurons (Bays, 2014, 2015). Given that memory has limited capacity it makes sense to distribute that capacity over items so as to minimize the distortion in the information that can be retrieved. More specifically, it makes sense to minimize the cost of errors. This is the basis of rate-distortion theory (Berger, 1971) which has been applied to successfully model data from visual STM tasks (Sims, 2015, 2016; Sims et al., 2012).

The case for resource models has been made predominantly for visual STM, where there continues to be an active debate as to whether there really is flexibility in resource allocation in visual STM (Fukuda, Awh, & Vogel, 2010; Ma, Husain, & Bays, 2014). Recently, Joseph et al. (2015) have argued that a resource account can also be applied to verbal STM. They showed that the precision of recall, as measured by participants' ability to adjust a probe stimulus to match one of the items to be remembered, decreased with memory load. However, this result is exactly what would be expected on the basis of any model that assumes that there is decay in STM such that representations become degraded over time - information is simply lost.

[3] Note that a bit is not a discrete quantum of information. A single event might be encoded using only a fraction of a bit which would reduce the uncertainly in the outcome of the event.

One way that the presence of chunks might influence memory in a resource model would be for resources to be distributed equally across chunks rather than items. This would make different predictions from a simple redintegration model as the presence of chunks in a list would now improve memory throughout the list. As the number of chunks in a list decreases, all chunks can be coded with greater precision. In an ideal communication system resources should also be distributed so as to reflect the statistical properties of the message and to ensure that less predictable or less easily recoverable parts of the message are encoded with greater precision. This is effectively a form of lossy compression where resources are distributed around list items so as to achieve a uniform improvement in memory performance or, in rate-distortion terms, the lowest cost. This is an example of how the efficiency of a communication channel can be improved by adapting to the statistics of the input (Shannon & Weaver, 1949).

Conclusion

Miller framed his original discussion in the context of information theory. He suggested that memory capacity was determined not by limitations on information capacity but by limits of the number of chunks that can be stored. This immediately raises the question of what is a chunk and what is it about a chunk that enables more items to be stored. The simplest view, consistent with Miller, Shiffrin and Nosofsky (1994), and Cowan and colleagues, is that a chunk in STM is simply a verbal label which can be used to retrieve the contents of the chunk from LTM. We suggest that the underlying limitation on STM capacity is information after all and that there are two factors that determine this capacity. The first is the nature of the representational vocabulary of the store. The second is how

efficiently we can utilize that vocabulary. Both are determined by the form of data compression involved.

Although we normally think of chunking as an explicit strategy, it might not be any different in principle from the automatic data compression that takes place in perception. For example, in speech perception the raw acoustic waveform is compressed into a representation that, in Barlow's (2001) terms, makes hidden redundancy manifest. That compressed code is then an ideal basis for representing speech information in a phonological short-term store with a limited information capacity. In effect, the phonological representations are already chunks that form an efficient compressed representation of a highly redundant acoustic waveform.

This constitutes a very general compression scheme that can achieve a high degree of compression with any message encoded in a listener's native language. However, if we ever need to remember or transmit only a subset of the possible messages, such as those made up of binary digits, it will be possible to use chunking to perform and additional level of compression. This second level of compression will often be time-consuming and effortful to apply. A further way of taking advantage of the presence of familiar chunks is to employ redintegration. Redintegration does not have the same costly overheads as it relies on the same processes that drive perception in general. It has the potential to capitalize on any accessible statistical knowledge of the environment.

Under most circumstances, chunking by compression and by redintegration will produce the same result - performance will improve when material can be represented as fewer and larger chunks. However, as we have seen, the two accounts can be teased apart by appropriately designed experiments. In the case of visual STM, Brady et al. (2009)

demonstrated that the presence of a chunk in a display can free up capacity and lead to improved memory for other items in the display. In the case of verbal STM, Norris and Kalm (2018) showed that in list containing chunks formed by pre-learned word triples and familiar singletons, the presence of more triples improved performance on both the triples themselves and the singletons. However, there is equally strong evidence for redintegration. When chunks were comprised of word pairs rather than triples, Norris and Kalm found that while overall performance improved when the lists contained more pairs, memory for the pairs and the triples within those lists remained constant. There was redintegration but no compression. Other evidence for redintegration comes from Botvinick and Bylsma (2005) who found that when participants had been familiarized with an artificial grammar they were less accurate at recalling sequences that were similar to grammatical sequences. Redintegration biased degraded memory representations towards more familiar patterns. The evidence reviewed here suggest that recall is influenced by both compression and redintegration. Performance is dominated by redintegration when the cost of recoding outweighs the benefits, and by compression when the benefits are greater; such as recoding binary to octal when there is no time pressure.

Our proposal is that any apparent chunk limit comes about from limits on how efficiently we can utilize the existing codes in a limited information capacity store. If monosyllabic words can potentially code 11 bits, then using the words ‘zero’ or ‘one’ to code a single binary digit is very inefficient. Coding three binary digits as one octal digit – or chunk - makes better used of the available information capacity, but comes at a cost. That is, chunking, or recoding, becomes a form of data compression. Chunking will be effective

because it makes better use of the available capacity, and not because the capacity itself is determined by the number of chunks (c.f. Brady et al., 2009).

By this account, capacity is determined by an interaction between the nature of the representational substrate of memory (samples, bits, phonemes, words) and the efficiency with which the input can be recoded in terms of those representations. With efficient use of a fine-grained level of representation (e.g. bits, or a continuous resource), capacity will appear to be determined by information. With less efficient use of a coarser-grained level of information, capacity will appear to be determined by the number of discrete chunks that can be stored. In a memory system that can store only words, capacity for chunks will be the same as capacity for words. In the absence of recoding at all, capacity will be determined by the number of items that can be stored. In all cases, capacity is a function of how effectively information in memory can be compressed.

References

Alvarez, G. A., & Cavanagh, P. (2004). The capacity of visual short-term memory is set both by visual information load and by number of objects. *Psychological Science, 15*(2), 106-111.

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, Mass: Harvard University Press.

Anderson, J. R., Bothell, D., Lebiere, C., & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38*, 341-380. doi:10.1006/jmla.1997.2553

Baddeley, A. D., & Hitch, G. (1974). Working memory. *Psychology of Learning and Motivation, 8*, 47-89.

Barlow, H. (2001). Redundancy reduction revisited. *Network (Bristol, England), 12*, 241-253.

Bays, P. M. (2014). Noise in neural populations accounts for errors in working memory. *Journal of Neuroscience, 34*(10), 3632-3645.

Bays, P. M. (2015). Spikes not slots: noise in neural populations limits working memory. *Trends in Cognitive Sciences, 19*(8), 431-438.

Bays, P. M., Catalao, R. F., & Husain, M. (2009). The precision of visual working memory is set by allocation of a shared resource. *Journal of Vision, 9*(10), 7-7.

Berger, T. (1971). *Rate distortion theory: A mathematical basis for data compression*. Englewood Cliffs, NJ: Prentice-Hall.

Bor, D., Duncan, J., Wiseman, R. J., & Owen, A. M. (2003). Encoding strategies dissociate prefrontal activity from working memory demand. *Neuron, 37*(2), 361-367.

Botvinick, M. M. (2005). Effects of domain-specific knowledge on memory for serial order. *Cognition, 97*, 135-151. doi:10.1016/j.cognition.2004.09.007

Botvinick, M. M., & Bylsma, L. M. (2005). Regularization in short-term memory for serial order. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 31*, 351-358. doi:10.1037/0278-7393.31.2.351

Bower, G. H. (1970). Organizational factors in memory. *Cognitive Psychology, 1*(1), 18-46.

Bower, G. H., & Winzenz, D. (1969). Group structure, coding, and memory for digit series. *Journal of Experimental Psychology, 80*(2p2), 1.

Brady, T. F., Konkle, T., & Alvarez, G. A. (2009). Compression in visual working memory: using statistical regularities to form more efficient memory representations. *Journal of Experimental Psychology: General, 138*(4), 487-502. doi:10.1037/a0016797

Brent, M. R., & Cartwright, T. A. (1996). Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition, 61*(1), 93-125.

Broadbent, D. E., & Broadbent, M. H. (1981). Articulatory suppression and the grouping of successive stimuli. *Psychological Research, 43*(1), 57-67.

Brown, G. D. A., & Hulme, C. (1995). Modeling item length effects in memory span: No rehearsal needed? *Journal of Memory and Language, 34*(5), 594.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology, 4*(1), 55-81.

Chekaf, M., Cowan, N., & Mathy, F. (2016). Chunk formation in immediate memory and how it relates to data compression. *Cognition, 155*, 96-107.

Chen, Z., & Cowan, N. (2005). Chunk limits and length limits in immediate recall: a reconciliation. *Journal of Experimental Psychology: Learning, Memory & Cognition, 31*(6), 1235-1249.

Chen, Z., & Cowan, N. (2009). Core verbal working-memory capacity: the limit in words retained without covert articulation. *Quarterly Journal of Experimental Psychology (2006), 62*, 1420-1429. doi:10.1080/17470210802453977

Cowan, N. (2001). The magical number 4 in short-term memory: a reconsideration of mental storage capacity. *Behavioral and Brain Sciences, 24*(1), 87-114; discussion 114-185.

Cowan, N., & Chen, Z. (2008). How chunks form in long-term memory and affect short-term memory limits. In M. Page & A. Thorn (Eds.), *Interactions between short-term and long-term memory in the verbal domain* (pp. 86-107). Hove, East Sussex, UK: Psychology Press.

Cowan, N., Chen, Z., & Rouder, J., N. (2004). Constant Capacity in an Immediate Serial-Recall Task. *Psychological Science, 15*, 634-640.

Cowan, N., Rouder, J. N., Blume, C. L., & Saults, J. S. (2012). Models of verbal working memory capacity: What does it take to make them work? *Psychological Review, 119*(3), 480.

Crawford, L. E., Huttenlocher, J., & Engebretson, P. H. (2000). Category effects on estimates of stimuli: Perception or reconstruction? *Psychological science, 11*(4), 280-284.

de Groot, A. D. (1946). *Het denken van den schaker: een experimenteel-psychologische studie*: Noord-Hollandsche Uitgevers Maatschappij.

de Marken, C. (1995). The unsupervised acquisition of a lexicon from continuous speech. Technical Report A.I. Memo No. 1558. AI Lab, MIT, Cambridge, Massachusetts.

Ericcson, K., Chase, W. G., & Faloon, S. (1980). Acquisition of a memory skill. *Science, 208*(4448), 1181-1182.

Ericsson, K. A., Delaney, P. F., Weaver, G., & Mahadevan, R. (2004). Uncovering the structure of a memorist's superior "basic" memory capacity. *Cognitive Psychology, 49*(3), 191-237.

Fougnie, D., Cormiea, S. M., Kanabar, A., & Alvarez, G. A. (2016). Strategic trade-offs between quantity and quality in working memory. *Journal of Experimental Psychology: Human Perception and Performance, 42*(8), 1231-1240. doi:doi:10.1037/xhp0000211

Frank, M. C., Goldwater, S., Griffiths, T. L., & Tenenbaum, J. B. (2010). Modeling human performance in statistical word segmentation. *Cognition, 117*(2), 107-125.

Frankish, C. (1985). Modality-Specific Grouping Effects in Short-Term-Memory. *Journal of Memory and Language, 24*(2), 200-209.

Fukuda, K., Awh, E., & Vogel, E. K. (2010). Discrete capacity limits in visual working memory. *Current Opinion in Neurobiology, 20*(2), 177-182.

Gathercole, S. E., Frankish, C. R., Pickering, S. J., & Peaker, S. (1999). Phonotactic influences on short-term memory. *Journal of Experimental Psychology: Learning Memory and Cognition, 25*, 84-95.

Glanzer, M., & Fleishman, J. (1967). The effect of encoding training on perceptual recall. *Perception & Psychophysics, 2*(12), 561-564.

Gobet, F. (1993). *A computer model of chess memory.* Paper presented at the Proceedings of 15th Annual Meeting of the Cognitive Science Society.

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

Gobet, F., Lloyd-Kelly, M., & Lane, P. C. (2016). What's in a name? The multiple meanings of "Chunk" and "Chunking". *Frontiers in psychology, 7*, 102.

Goldwater, S., Griffiths, T. L., & Johnson, M. (2009). A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition, 112*(1), 21-54.

Grünwald, P. D. (2007). *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*: The MIT Press.

Grünwald, P. D., Myung, I. J., & Pitt, M. A. (2005). *Advances in minimum description length: Theory and applications*: MIT press.

Hayes, J. R. M. (1952). *Memory span for several vocabularies as a function of vocabulary size*. Retrieved from Cambridge, Mass:

Hebb, D. O. (1961). Distinctive features of learning in the higher animal. In J. F. Delafresnaye (Ed.), *Brain Mechanisms and Learning* (pp. 37-46). Oxford: Blackwell.

Hemmer, P., & Steyvers, M. (2009a). A Bayesian account of reconstructive memory. *Topics in Cognitive Science, 1*(1), 189-202.

Hemmer, P., & Steyvers, M. (2009b). Integrating episodic memories and prior knowledge at multiple levels of abstraction. *Psychonomic Bulletin & Review, 16*(1), 80-87.

Henson, R. N. (1998). Short-term memory for serial order: the Start-End Model. *Cognitive Psychology, 36*, 73-137. doi:10.1006/cogp.1998.0685

Henson, R. N., Burgess, N., & Frith, C. D. (2000). Recoding, storage, rehearsal and grouping in verbal short-term memory: an fMRI study. *Neuropsychologia, 38*, 426-440.

Hitch, G. J., Burgess, N., Towse, J. N., & Culpin, V. (1996). Temporal grouping effects in immediate recall: A working memory analysis. *Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology, 49*(1), 116-139.

Huang, L., & Awh, E. (2018). Chunking in working memory via content-free labels. *Scientific reports, 8*(1), 23.

Hulme, C., Maughan, S., & Brown, G. D. (1991). Memory for familiar and unfamiliar words: Evidence for a long-term memory contribution to short-term memory span. *Journal of Memory and Language, 30*, 685-701.

Hulme, C., Roodenrys, S., Brown, G., & Mercer, R. (1995). The Role of Long-Term-Memory Mechanisms in Memory Span. *British Journal of Psychology, 86*, 527-536. doi:10.1037/0278-7393.23.5.1217

Hulme, C., Roodenrys, S., Schweickert, R., Brown, G. D. A., Martin, S., & Stuart, G. (1997). Word-frequency effects on short-term memory tasks: Evidence for a redintegration process in immediate serial recall. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 23*(5), 1217.

Huttenlocher, J., Hedges, L. V., & Duncan, S. (1991). Categories and particulars: Prototype effects in estimating spatial location. *Psychological Review, 98*(3), 352.

Huttenlocher, J., Hedges, L. V., & Vevea, J. L. (2000). Why do categories affect stimulus judgment? *Journal of Experimental Psychology: General, 129*(2), 220.

Johnson, E. K., & Jusczyk, P. W. (2001). Word segmentation by 8-month-olds: When speech cues count more than statistics. *Journal of Memory and Language, 44*(4), 548-567.

Jones, G. (2016). The influence of children's exposure to language from two to six years: The case of nonword repetition. *Cognition, 153*, 79-88.

Jones, G., Gobet, F., Freudenthal, D., Watson, S. E., & Pine, J. M. (2014). Why computational models are better than verbal theories: the case of nonword repetition. *Developmental Science, 17*(2), 298-310.

Jones, G., & Macken, B. (2018). Long-term associative learning predicts verbal short-term memory performance. *Memory & Cognition, 46*(2), 216-229.

Jones, G., & Macken, W. J. (2015). Questioning short-term memory and its measurement: Why digit span measures long-term associative learning. *Cognition, 144*, 1-13.

Jones, T., & Farrell, S. (2018). Does syntax bias serial order reconstruction of verbal short-term memory? *Journal of Memory and Language, 100*, 98-122.

Joseph, S., Iverson, P., Manohar, S., Fox, Z., Scott, S. K., & Husain, M. (2015). Precision of working memory for speech sounds. *The Quarterly Journal of Experimental Psychology, 68*(10), 2022-2040.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: individual differences in working memory. *Psychological Review, 99*(1), 122.

Kleinberg, J., & Kaufman, H. (1971). Constancy in short-term memory: Bits and chunks. *Journal of Experimental Psychology, 90*(2), 326.

Larkin, J., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Expert and novice performance in solving physics problems. *Science, 208*(4450), 1335-1342.

Lewandowsky, S., & Farrell, S. (2000). A redintegration account of the effects of speech rate, lexicality, and word frequency in immediate serial recall. *Psychological Research, 63*(2), 163-173.

Ma, W. J., Husain, M., & Bays, P. M. (2014). Changing concepts of working memory. *Nature Neuroscience, 17*(3), 347-356.

Majerus, S., Van der Linden, M., Mulder, L., Meulemans, T., & Peters, F. (2004). Verbal short-term memory reflects the sublexical organization of the phonological language

network: Evidence from an incidental phonotactic learning paradigm. *Journal of Memory and Language, 51*, 297-306. doi:10.1016/j.jml.2004.05.002

Mathy, F., & Feldman, J. (2012). What's magic about magic numbers? Chunking and data compression in short-term memory. *Cognition, 122*, 346-362. doi:10.1016/j.cognition.2011.11.003

McLean, R., & Gregg, L. (1967). Effects of induced chunking on temporal aspects of serial recitation. *Journal of Experimental Psychology, 74*(4p1), 455.

Miller, G. A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63*, 81-97.

Newell, A. (1994). *Unified theories of cognition*: Harvard University Press.

Norris, D. (2006). The Bayesian reader: explaining word recognition as an optimal Bayesian decision process. *Psychological Review, 113*(2), 327.

Norris, D. (2017). Short-Term Memory and Long-Term Memory are Still Different. *Psychological Bulletin, 143*(9), 992-1009.

Norris, D. G., & Kalm, K. (2018, May 9). Chunking and redintegration in verbal short-term memory. http://doi.org/10.17605/OSF.IO/SM736

Norris, D., & McQueen, J. M. (2008). Shortlist B: A Bayesian model of continuous speech recognition. *Psychological Review, 115*(2), 357.

Orbán, G., Fiser, J., Aslin, R. N., & Lengyel, M. (2008). Bayesian learning of visual chunks by human observers. *Proceedings of the National Academy of Sciences of the United States of America, 105*, 2745-2750. doi:10.1073/pnas.0708424105

Orhan, A. E., & Jacobs, R. A. (2013). A probabilistic clustering theory of the organization of visual short-term memory. *Psychological Review, 120*(2), 297.

Orhan, A. E., Sims, C. R., Jacobs, R. A., & Knill, D. C. (2014). The adaptive nature of visual working memory. *Current Directions in Psychological Science, 23*(3), 164-170.

Poirier, M., & Saint-Aubin, J. (1996). Immediate serial recall, word frequency, item identity and item position. *Canadian Journal of Experimental Psychology, 50*(4), 408-412.

Pollack, I. (1953). Assimilation of sequentially encoded information. *The American journal of psychology, 66*(3), 421-435.

Pollack, I., & Johnson, L. B. (1965). Memory-span with efficient coding procedures. *The American Journal of Psychology, 78*(4), 609-614.

Redlich, A. N. (1993). Redundancy reduction as a strategy for unsupervised learning. *Neural Computation, 5*, 289-304.

Rieke, F., Bodnar, D., & Bialek, W. (1995). Naturalistic stimuli increase the rate and efficiency of information transmission by primary auditory afferents. *Proceedings of the Royal Society of London B: Biological Sciences, 262*(1365), 259-265.

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, 35*(7), 1352-1389.

Roodenrys, S., & Miller, L. M. (2008). A constrained Rasch model of trace redintegration in serial recall. *Memory & Cognition, 36*, 578-587. doi:10.3758/MC.36.3.578

Ryan, J. (1969a). Grouping and short-term memory: different means and patterns of grouping. *The Quarterly Journal of Experimental Psychology, 21*(2), 137-147.

Ryan, J. (1969b). Temporal grouping, rehearsal and short-term memory. *The Quarterly Journal of Experimental Psychology, 21*(2), 148-155. Retrieved from

Schwartz, M., & Bryden, M. P. (1971). Coding factors in the learning of repeated digit sequences. *Journal of Experimental Psychology, 87*(3), 331-334.

Schweickert, R. (1993). A multinomial processing tree model for degradation and redintegration in immediate recall. *Memory & Cognition, 21*(2), 168-175.

Servan-Schreiber, E., & Anderson, J. R. (1990). Learning artificial grammars with competitive chunking. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16*(4), 592.

Shannon, C. E., & Weaver, W. (1949). *The mathematical theory of communication*. Urbana: University of Illinois Press.

Shiffrin, R. M., & Nosofsky, R. M. (1994). Seven plus or minus two: A commentary on capacity limitations. *Psychological Review, 101*(2), 357-361. doi:doi.org/10.1037

Shiffrin, R. M., & Steyvers, M. (1997). A model for recognition memory: REM-retrieving effectively from memory. *Psychonomic Bulletin & Review, 4*, 145-166. doi:10.3758/BF03209391

Simon, H. A. (1974). How big is a chunk? *Science, 183*(4124), 482-488.

Simon, H. A., & Feigenbaum, E. A. (1964). An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning. *Journal of Verbal Learning and Verbal Behavior, 3*(5), 385-396.

Sims, C. R. (2015). The cost of misremembering: Inferring the loss function in visual working memory. *Journal of Vision, 15*(3), 1-27. doi:10.1167/15.3.2

Sims, C. R. (2016). Rate–distortion theory and human perception. *Cognition, 152*, 181-198. doi:10.1016/j.cognition.2016.03.020

Sims, C. R., Jacobs, R. A., & Knill, D. C. (2012). An ideal observer analysis of visual working memory. *Psychological Review, 119*(4), 807.

Stuart, G. P., & Hulme, C. (2009). Lexical and semantic influences on immediate serial recall: A role for redintegration. In A. Thorne & M. Page (Eds.), *Interactions between short-term and long-term memory in the verbal domain* (pp. 157-176). Hove, UK: Psuchology Press.

Thorn, A. S., & Gathercole, S. E. (1999). Language-specific knowledge and short-term memory in bilingual and non-bilingual children. *The Quarterly Journal of Experimental Psychology: Section A, 52*(2), 303-324.

Thorn, A. S., Gathercole, S. E., & Frankish, C. R. (2002). Language familiarity effects in short-term memory: the role of output delay and long-term knowledge. *The Quarterly Journal of Experimental Psychology: Section A, 55*(4), 1363-1383.

Wickelgren, W. A. (1964). Size of rehearsal group and short-term memory. *Journal of Experimental Psychology, 68*(4), 413.

Winzenz, D. (1972). Group structure and coding in serial learning. *Journal of Experimental Psychology, 92*(1), 8.

Xu, J., & Griffiths, T. L. (2010). A rational analysis of the effects of memory biases on serial reproduction. *Cognitive Psychology, 60*(2), 107-126.

Ye, N., & Salvendy, G. (1994). Quantitative and qualitative differences between experts and novices in chunking computer software knowledge. *International Journal of Human-Computer Interaction, 6*(1), 105-118.

Zipf, G. K. (1935). *The psycho-biology of language: an introduction to dynamic philology*. Boston: Houghton Miffin.

Box 1

Basic concepts

Lossless compression.
Compress in such a way that the original message can be fully reconstructed.
e.g. zip files, or FLAC audio files.
The strategy of recoding binary digits as octal digits is a form of lossless compression if the message is known to contain only binary digits.

Deletion code: Form a new code for a chunk by deleting redundant information e.g. *brick, hat -> brick*.

Lossless compression without chunking:
Huffman code - replace more frequent items with shorter codes (c.f. Zipf's law)

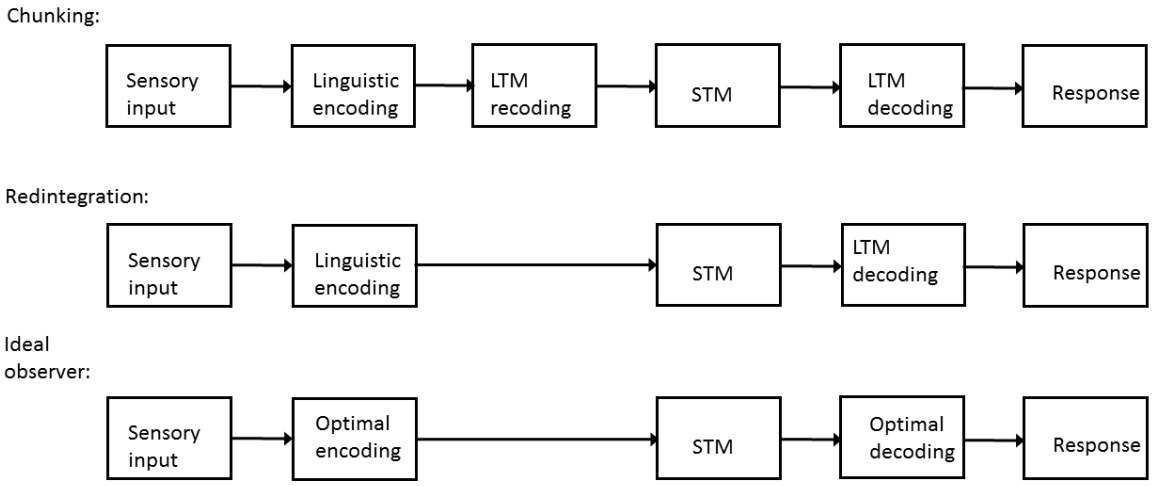
Lossy compression
Compression which does not permit the original message can be fully reconstructed.
e.g. MP3, JPEG
Most of perception can be thought of as lossy compression. e.g. Perceiving an acoustic waveform as a word constructs a compressed representation from which the original waveform cannot be reconstructed.

Redintegration:
Reconstructing a degraded trace in STM with reference to information in LTM.
$p(chunk_i|x) \propto p(x|chunk_i) \cdot p(chunk_i)$
Where $p(chunk_i|x)$ is the probability of recalling the items forming $chunk_i$ given some, possibly degraded, information in STM, x, , and $p(chunk_i)$ is the prior probability of that chunk. The greater the probability of the chunk, the more likely it is to be recalled correctly.

Minimum description length. (Rissanen, 1978)
A metric for comparing compression methods by computing the amount of information required to represent the entire message, which may consist of a coded message and a codebook.

Self-information
The amount of information needed to code each item in a message as a function of its probability:
$I = -\log_2(p_i)$

Figure 1



Schematic comparison of chunking, redintegrationa and ideal-observer models.