Fully Distributed Representation

Pentti Kanerva

RWCP Neuro SICS Laboratory
Real World Computing Partnership
Swedish Institute of Compute Science
e-mail: kanerva@sics.se

Abstract

A fully distributed representation based on the binary spatter code is described. It is shown how the information of a conventional record with fields is encoded into a long random bit string, or a holistic record, that has no fields, and how the fields are extracted from the holistic record. It is argued that holistic representation should be used in modeling high-level mental functions.

1. INTRODUCTION

Local representation—records with fields—so dominates our computing practices that we are hardly aware of alternative representations, or even the need for any. By representation is meant, simply, how information is laid out in bits in some physical medium, for example in a computer memory or a neural net. With local representation, the meaning of a bit—what a bit refers to—is tied to its location. To decide what a bit pattern is, we must therefore know where it comes from. This creates the need for a system that keeps track of where information is located. A computer program is such a system.

The search for distributed representations (e.g., Anderson, 1995; Hinton, 1990; Hinton et al., 1986; Pollack, 1990; Smolensky, 1990; Touretzky & Geva, 1987; Touretzky, 1990) has grown out of the realization that traditional representation, as used in computers, is not only artificial but that it may actually hinder the development of the kind of computing that makes brains intelligent. One reason would be that traditional representation relies on a program to interpret it, and that intelligent programs of the traditional kind are extremely difficult to find with automated learning methods because programs are unstable. A superficially minor change—an incorrect bit or a misplaced symbol—can wreck an otherwise good program. Learning methods that rely on incremental improvements cope poorly with instability, and good programs are so rare among all possible programs that finding them by chance is hopeless.

Written as a tutorial, this paper demonstrates with simple examples a fully distributed alternative to local representation, so as to make the ideas accessible. Properties of the representation are discussed, and pointers to past and future research are given. For the benefit of those already familiar with distributed representation it is pointed out that the present system is related to Plate's (1994) Holographic Reduced Representation and is described in the same terms.

2. RECORDS WITH FIELDS

Local representation stores information in *records* with *fields.* Figure 1 shows some examples. A binary attribute- or feature-vector is the simplest example. Such a vector for N attributes is then a record with N one-bit fields. Figure 1a shows unary encoding of the English alphabet with such vectors. Unary encoding and its real-valued counterpart are common in statistical classifiers (e.g., artificial neural nets), with each class having its own output variable. Such codes are easy for us to interpret but their use of bits is wasteful.

Figure 1b shows a record for a name made of 12 five-bit fields for the letters. The representation of a letter within a field is now distributed, as no particular bit will tell whether the letter is an A, for example. In Figure 1c such a record is used as the name field of another record that has fields also for sex and age.

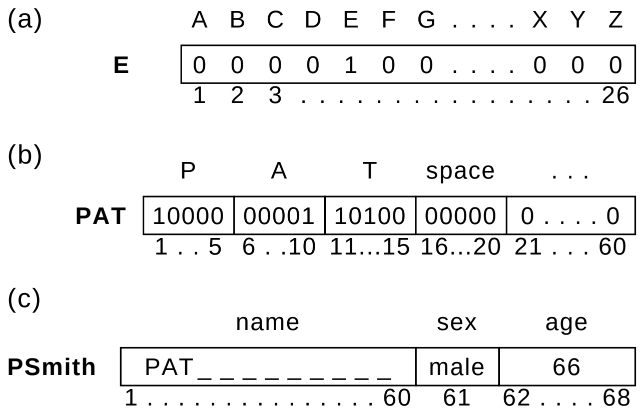

Figure 1. Local representation: records with fields.

Figure 2 makes the point that two records that differ from each other minimally can mean very different things. It demonstrates that traditional data representation is unstable in the same sense as traditional programs are. Figure 3 shows that the same information can take very different forms.

3. DISTRIBUTING THE RECORD

In this section we see how the information of a traditional record, the three-field record of Figure 1c, can be encoded into a binary vector without fields. Each field will be distributed fully, over the entire vector, so that each bit contains some information about every field. The resulting record is called *holographic* or *holistic*.

Codewords or -vectors. The holistic record is built of codewords for field names (variables, roles) and for values that occupy the fields (fillers). All codewords are long, random bit strings or binary N-vectors; N = 10,000 will be used in our examples. The N bit positions or coordinates or columns (as in a table where the vectors are the rows) of a codeword are independent of each other, and 0s and 1s are equally probable, $p = \Pr\{1\} = 0.5$. To encode the record for Pat Smith, one such codeword is needed for each of ‘name’, ‘Pat’, ‘sex’, ‘male’, ‘age’, and ‘66’. The codewords are written in boldface: **name**, **Pat**, **sex**, **male**, **age**, and **66**.

Binding and chunking. A holistic record is composed in two steps called binding and chunking. Binding encodes a field, and chunking combines the fields into a record. *Binding* is done with coordinatewise Boolean Exclusive-OR (XOR, ⊗), so that ‘name = Pat’ is represented by the 10,000-bit codeword **name**⊗**Pat**. This corresponds to storing ‘Pat’ in the name field of a traditional record. Similarly, **sex**⊗ **male** and **age**⊗**66** encode the other two fields.

The encoded fields are *chunked* into a holistic record according to the majority rule. Each bit of the record will be a 0 or a 1 according to which of them appears more often in that position, or column, in the (three) vectors that are being chunked. When the number of chunked vectors is even and there are ties, they are broken at random with probability 1/2. The composed record also has N bits, with 0s and 1s equally probable, and thus it can, in turn, be assigned as a value to a field (i.e., bound to a variable) and chunked into further N-bit records. This property makes the code *recursive*.

The majority rule can be realized as a thresholded sum: by thresholding the columnwise sums at half the number of chunked vectors. The holistic record for Pat Smith—the encoding of ‘name = Pat & sex = male & age = 66’—can then be expressed as

$$\mathbf{PSmith} = [\mathbf{name}\otimes\mathbf{Pat} + \mathbf{sex}\otimes\mathbf{male} + \mathbf{age}\otimes\mathbf{66}]$$

where the brackets [...] indicate thresholding. The vectors that are combined by chunking are also called *parts*. To break ties when the number of parts is even, we can chunk a random vector **R** with them (that possibly is a function of the parts).

Chunking can be compared to creating a *pointer* to a conventional record (see Fig. 3), which then represents the record. This is how symbolic (list) structures are constructed. The main similarity between the two is uniformity: all chunked records have the same number of bits, N, and all pointers have the same number of bits, L. One major difference is that N is much larger than L (usually $L < 30$), and the other, which is related to the first, is that a conventional pointer hides its relation to the contents of the record, whereas chunking leaves the parts visible.

Visibility of Parts. When K codewords are chunked, the resulting codeword bears more than chance resemblance to its constituent parts. The smaller K is, the greater the similarity. If $K = 3$ and if **A**, **B**, and **C** are random and independent and $\mathbf{X} = [\mathbf{A} + \mathbf{B} + \mathbf{C}]$, how close is **X** to **A**, or **PSmith** to **name**⊗**Pat**? The expected (Hamming) distance, relative to N, is the probability that the corresponding bits of **X** and **A** differ, $\delta(\mathbf{X}, \mathbf{A}) = \Pr\{X_n \neq A_n\}$. If a bit

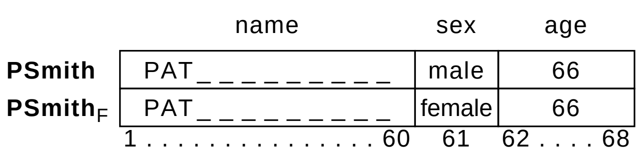

Figure 2. Two very different Pat Smiths, two very similar bit strings.

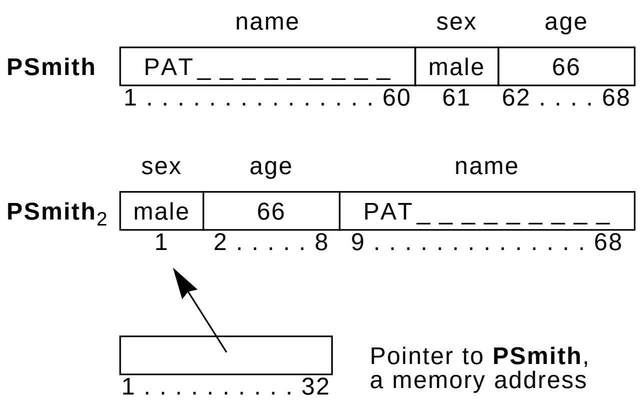

Figure 3. Three representations of **PSmith**, three very different bit strings.

of **A** is a 0 and the same bit of **X** is a 1, it must be a 1 also in **B** and **C** to constitute a majority. This happens with probability 0.25, and it is the same when a bit of **A** is a 1. So the bits of **X** and **A** differ with probability 0.25, and thus **X** is much closer to **A** (and to **B** and to **C**), and **PSmith** to **name**⊗**Pat**, than what it would be by chance: $d(\mathbf{X}, \mathbf{A}) = 0.25 \pm 0.0043$, whereas $d(\mathbf{A}, \mathbf{B}) = 0.5 \pm 0.005$ by chance (getting $d \le 0.25$ by chance is as unlikely as getting no more than 2,500 heads in 10,000 tosses of a coin). The standard deviation of distance is based on the binomial distribution for N = 10,000 and δ = 0.25, 0.5 and it is given by $\sigma = \sqrt{(\delta(1-\delta))/N}$.

For $K \neq 3$ the distance is the following. If K is even, we first add in a random vector **R** for a tie-breaker and write it as $\mathbf{X} = [\mathbf{A} + \mathbf{B} + \ldots + \mathbf{D}\ (+\ \mathbf{R})]$. Now K is odd, and the expected distance is

$$\delta(\mathbf{X}, \mathbf{A}) = \frac{1}{2} - \binom{K-1}{(K-1)/2} \Big/ 2^K$$

which is approximately $0.5 - 0.4/\sqrt{K - 0.44}$ (this approximation is an improvement over one based on Stirling's factorial formula when K = 3, 5, 7, …, 49). The expected distance is shown in Figure 4. The thing to notice is that d need not be far from 1/2 to be significantly different from it when N is large since its standard deviation is so close to zero. It is in this statistical sense that a chunk resembles its parts and differs from unrelated codewords.

The distance δ is related to the correlation coefficient (normalized covariance) by $\rho = 1 - 2\delta$ (the relation is linear also when $p \neq 0.5$, although it is more complicated). The right margin of Figure 4 is labeled with these correlations. Although the correlation between a chunk and its parts is low, it is very significant when the code is wide (N > 1,000) and the parts are few (K < 15).

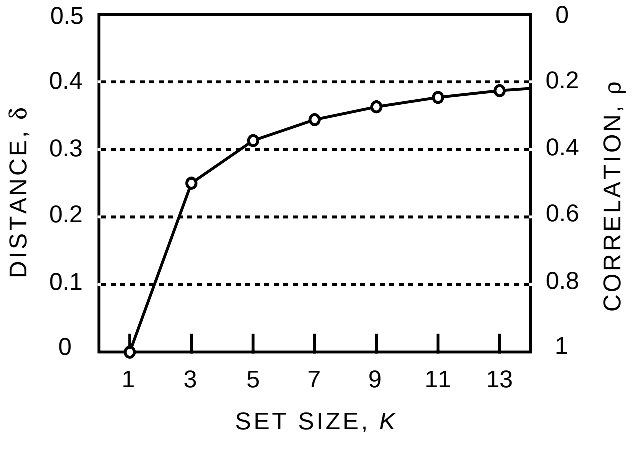

Figure 4. Distance and correlation between a holistic record and its parts as a function of the number of parts (set size) K.

Decoding. When XOR is used for binding, as in **name**⊗**Pat**, it is used also for "unbinding" or decoding. To extract **Pat** from an encoded name-field, we XOR with **name**:

$$\mathbf{name} \otimes (\mathbf{name} \otimes \mathbf{Pat}) = (\mathbf{name} \otimes \mathbf{name}) \otimes \mathbf{Pat} = \mathbf{Pat}$$

because XOR is associative and is its own inverse function. This is called *probing*.

Probing with **name** is used also for extracting **Pat** from the chunked record **PSmith**. Because **PSmith** resembles **name**⊗**Pat**, **name**⊗**PSmith** resembles **name**⊗(**name**⊗**Pat**), which is **Pat**. In fact, the distance d(**name**⊗**PSmith**, **Pat**) = d(**PSmith**, **name**⊗**Pat**) = 0.25 ± 0.0043, because XORing with the same vector, **name**, leaves the distances between vectors unchanged. In the following we will write **name**⊗**PSmith** as **Pat′**.

Clean-up Memory. We can think of **Pat′** as a noisy version of **Pat** that needs cleaning up. For that purpose the system has a clean-up memory that keeps track of all valid codewords—ones that have been defined so far—and that takes noisy codewords as inputs and produces noise-free codewords as outputs. (The valid codewords mentioned up to now are **name**, **Pat**, **sex**, **male**, **age**, **66**, and **PSmith**, whereas **name**⊗**Pat**, **sex**⊗**male**, and **age**⊗**66** need not be stored in the clean-up memory.)

We still need to assure that the decoding is unambiguous. Although **Pat′** is close to **Pat**, it may also be close to other valid codewords in the clean-up memory. So let us look closely at the structure of **Pat′**:

$$\begin{aligned}\mathbf{Pat'} &= \mathbf{name} \otimes \mathbf{PSmith} \\ &= \mathbf{name} \otimes [\mathbf{name} \otimes \mathbf{Pat} + \mathbf{sex} \otimes \mathbf{male} + \mathbf{age} \otimes \mathbf{66}]\end{aligned}$$

which equals

$$[\mathbf{name} \otimes (\mathbf{name} \otimes \mathbf{Pat}) + \mathbf{name} \otimes (\mathbf{sex} \otimes \mathbf{male}) + \mathbf{name} \otimes (\mathbf{age} \otimes \mathbf{66})]$$

In other words, the XOR distributes over the chunked parts.

To see why $\mathbf{a} \otimes \mathbf{X} = \mathbf{a} \otimes [\mathbf{A} + \ldots + \mathbf{D}\ (+\ \mathbf{R})] = [\mathbf{a} \otimes \mathbf{A} + \ldots + \mathbf{a} \otimes \mathbf{D}\ (+\ \mathbf{a} \otimes \mathbf{R})]$, first consider the places (i.e., bit positions or columns) where **a** has a 0; in those places XORing with **a** has no effect on **X** or on any of its parts **A**, …, **D**, or **R**, so that in those places distributivity holds trivially. Secondly, wherever **a** has a 1 it flips the bits of **X**, and we get the same result by flipping those same bits in **A**, …, **D**, and **R** and then applying the majority rule—wherever 0s were in the majority now 1s are, and vice versa.

The above expression gives us

Pat′ = [**Pat** + **name**⊗**sex**⊗**male** + **name**⊗**age**⊗**66**]

which shows that, due to its structure, **Pat′** resembles also the vectors **name**⊗**sex**⊗**male** and **name**⊗**age**⊗ **66** (δ = 0.25). But these are not valid codewords and therefore they are not in the clean-up memory. They act merely as random noise (their distance to valid codewords is 0.5 ± 0.005).

Figure 5 summarizes holistic encoding of a conventional record of Figure 1c and extracting a "field" from the holistic record.

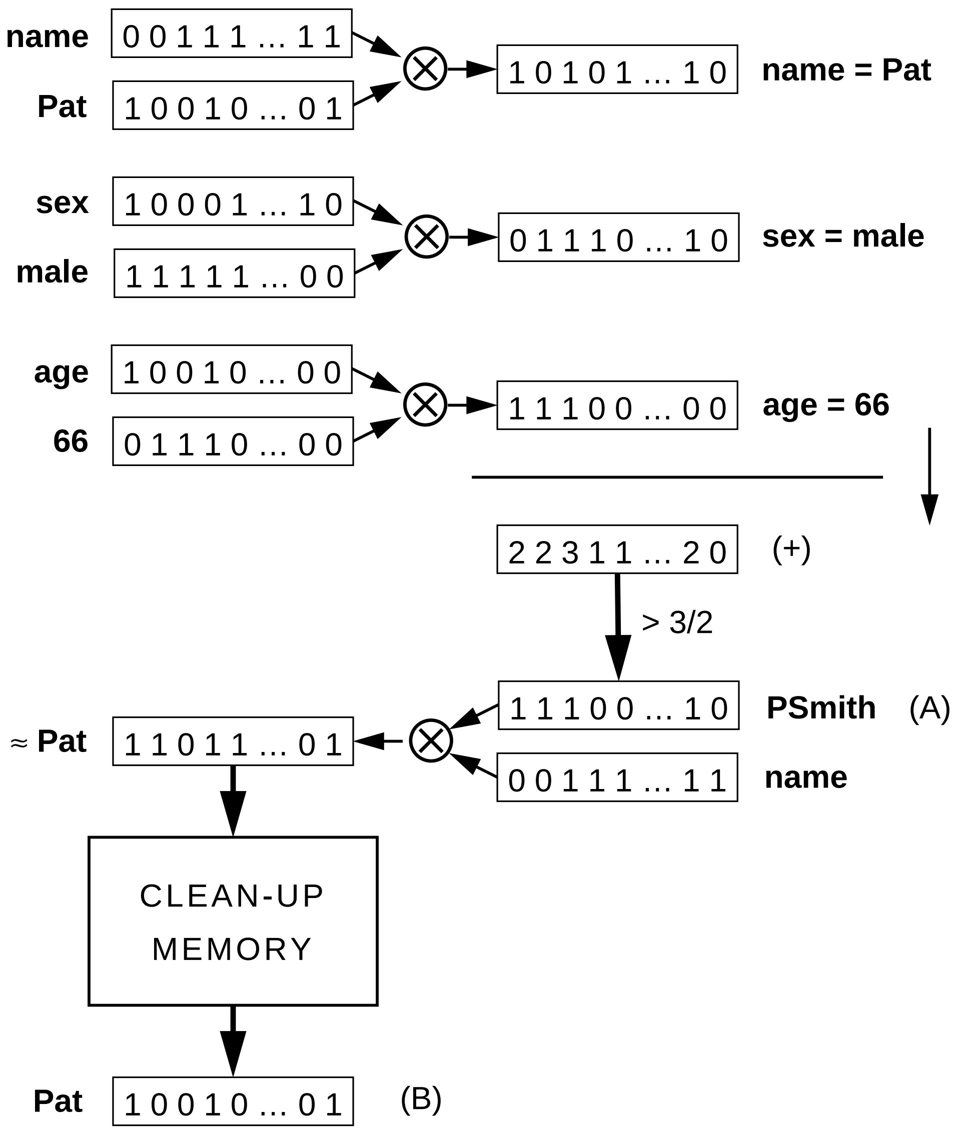

Figure 5. Holistic encoding and decoding. 'name = Pat & sex = male & age = 66' has been composed into the holistic record **PSmith** (A), and then the name has been extracted from it (B). Codewords are random 10,000-bit strings, ⊗ is bitwise Boolean XOR, and bitwise sums (+) are thresholded at 3/2, constituting the majority rule. '≈ **Pat**' means that the result is approximate but close enough for the Clean-up Memory to identify it.

4. PROCESSING EXAMPLES

The following examples provide insight into holistic representation. Let us assume that we have four records with identical structure, as shown in Table 1. They give rise to 13 codewords in the clean-up memory, namely, **name**, **Pat**, **Lee**, **sex**, **male**, **female**, **age**, **33**, **66**, **PF3** = [**name**⊗**Pat** + **sex**⊗**female** + **age**⊗**33**], **PM6** = [**name**⊗**Pat** + **sex**⊗**male** + **age**⊗**66**], **LF6** = [**name**⊗**Lee** + **sex**⊗**female** + **age**⊗**66**], and **LM3** = [**name**⊗**Lee** + **sex**⊗**male** + **age**⊗**33**].

We now ask the following question: "What is the age of Lee who is a female?" This refers to **LF6**, and we look for **66**.

Case 1. In this case we assume that the codewords **name**, **Lee**, **sex**, **female**, and **age** are known to us, but **LF6** is unknown even if it is in the clean-up memory.

Solution 1. We can first approximate **LF6** by replacing the unknown **age**⊗**66** with a random vector **R**, giving **LF**x = [**name**⊗**Lee** + **sex**⊗**female** + **R**]. It contains 25% noise relative **LF6**, 37.5% relative to **PF3** and to **LM3**, and 50% relative to the other ten codewords. The distances with standard deviations for N = 10,000 are d(**LF**x, **LF6**) = 0.25 ± 0.0043; d(**LF**x, **PF3**), d(**LF**x, **LM3**) = 0.375 ± 0.0048; and d(**LF**x, other) = 0.5 ± 0.005. Because the standard deviations are so small, cleaning up **LF**x will find the correct **LF6** with very high probability. Thereafter we can probe **LF6** with **age** and clean the result to **66**. We will write that as **age**⊗**LF6** → clean-up (0.25) → **66**, where the 0.25 means that the amount of noise to be cleaned up is 25%.

Case 2. In this case **Pat**, **male**, **PM6**, **Lee**, **female**, and **age** are known, but **name**, **sex**, and **LF6** are unknown.

Solution 2c (*c* for 'conventional'). We can find **name** and **sex** by **Pat**⊗**PM6** → clean-up (0.25) → **name** and **male**⊗**PM6** → clean-up (0.25) → **sex**. From here on we can apply Solution 1.

Solution 2u (*u* for 'unconventional'). Here we use the correspondences **Pat** ↔ **Lee** and **male** ↔ **female** to form the mapping **T** = [**Pat**⊗**Lee** + **male**⊗**female** + **R**] and with it map **PM6** as a whole: **PM6**⊗**T** → clean-up (0.375) → **LF6**. The other two distances that differ from 0.5 are significantly greater than 0.375, namely, d(**PM6**⊗**T**, **PF3**), d(**PM6**⊗**T**, **LM3**) = 0.44 ± 0.005. Finally, **age**⊗**LF6** → clean-up (0.25) → **66**, as above.

Solution 2u provides an opportunity to look at the inner workings of holistic representation, its underlying algebra. **PM6**⊗**T** is first written out as

$$\mathbf{PM6}\otimes\mathbf{T} = \mathbf{PM6}\otimes[\mathbf{Pat}\otimes\mathbf{Lee} + \mathbf{male}\otimes\mathbf{female} + \mathbf{R}]$$

Table 1
Four Records for Holistic Encoding

	name	sex	age
PF3	Pat	female	33
PM6	Pat	male	66
LF6	Lee	female	66
LM3	Lee	male	33

which, by distributivity, equals

$$[\mathbf{PM6}\otimes\mathbf{Pat}\otimes\mathbf{Lee} + \mathbf{PM6}\otimes\mathbf{male}\otimes\mathbf{female} + \mathbf{PM6}\otimes\mathbf{R}]$$

Substituting [**name**⊗**Pat** + **sex**⊗**male** + **age**⊗**66**] for **PM6** and applying distributivity a second time gives

$$\begin{aligned}\mathbf{PM6}\otimes\mathbf{T} = [[&\mathbf{name}\otimes\mathbf{Pat}\otimes\mathbf{Pat}\otimes\mathbf{Lee}\\ &+ \mathbf{sex}\otimes\mathbf{male}\otimes\mathbf{Pat}\otimes\mathbf{Lee}\\ &+ \mathbf{age}\otimes\mathbf{66}\otimes\mathbf{Pat}\otimes\mathbf{Lee}]\\ &+ [\mathbf{name}\otimes\mathbf{Pat}\otimes\mathbf{male}\otimes\mathbf{female}\\ &+ \mathbf{sex}\otimes\mathbf{male}\otimes\mathbf{male}\otimes\mathbf{female}\\ &+ \mathbf{age}\otimes\mathbf{66}\otimes\mathbf{male}\otimes\mathbf{female}]\\ &+ \mathbf{PM6}\otimes\mathbf{R}]\end{aligned}$$

which simplifies to

$$\begin{aligned}\mathbf{PM6}\otimes\mathbf{T} = [[&\mathbf{name}\otimes\mathbf{Lee} + \mathbf{R}_1 + \mathbf{R}_2]\\ &+ [\mathbf{R}_3 + \mathbf{sex}\otimes\mathbf{female} + \mathbf{R}_4] + \mathbf{R}_5]\end{aligned}$$

where the $\mathbf{R}_i$ stand for vectors that act as random noise, as nothing corresponding to them is stored in the clean-up memory. From this last expression for **PM6**⊗**T** we see why mapping with **T** takes **PM6** near **LF6**: two of its parts are the same as in **LF6**, with some noise added. The breakdown of **LF**x in Solution 1 is similar except that the two known parts there are noise-free. In that case the result is 25% noisy compared to the present 37.5%.

Case 3. Now **33**, **PF3**, and **LF6** are known. The codewords **33** and **PF3** would allow us to find **age**, and with it we could probe **LF6**, but it is instructive to look at a less conventional one-step solution.

Solution 3a (*a* for 'analogous' [and also for 'ambiguous']). Think of a 33-year old Pat (**PF3**) asking Lee (**LF6**) the following question: "I'm 33; what are you?" Of course Lee could answer "I'm female," but that would seem unnatural whereas "I'm 66" seems natural. We will see how the holistic representation favors the latter answer over the former.

Since **A** = **33**⊗**PF3** ≈ **age** and **age**⊗**LF6** ≈ **66**, we can try **A**⊗**LF6** directly without first recovering **age**. We get that **A**⊗**LF6** is equally far from **66** and **33**—d(**A**⊗**LF6**, **66**), d(**A**⊗**LF6**, **33**) = 0.375 ± 0.0048—

and it is 0.5 away from all other valid codewords. So although we do not find **66** unambiguously, a true but anomalous answer such as “I’m female” is not even suggested. The expression (**33**⊗**PF3**)⊗**LF6** is easily broken down algebraically, in the style of Solution 2u, and the breakdown shows how the two equally likely answers emerge.

Comments on the Examples. The above examples give cause for several comments.

1. The four records in Table 1 were chosen so that none of them is determined by one attribute alone, so although the records are few, they are nontrivial.

2. The records need not have identical structure. For example, records that do not encode age are automatically ignored, and records that encode all three attributes plus additional ones would give somewhat noisier results but would not confuse matters otherwise. Also, the codewords used in encoding the records can be noisy; there merely will be more noise to clean up. Notice, also, that the bit pattern of a conventional record depends on the order in which the fields appear (cf. Fig. 3), whereas the order becomes irrelevant when the information is encoded into a holistic record.

3. The examples are not meant to suggest that traditional data bases should be encoded in this holistic manner. These examples work because the records have few fields. If the fields are too many, extracting them from a holistic record is unreliable. However, holistic records could supplement traditional data bases, but how that should be done remains to be researched.

4. Solutions 2u and 3a point to a fundamentally new way of computing, namely, by holistic mapping, which is a mapping between points of a very-high-dimensional space. More is said about it below under *Beyond Correlation*.

5. SUMMARY AND DISCUSSION

Information with structure, such as language with its grammatical structure or concepts built of other concepts by some logic, has traditionally been the domain of symbolic representation. Artificial neural nets as mathematical models of the brain need to handle such information. Out of this need has grown holistic representation.

Holistic representation is fundamentally distributed whereas traditional representation, with its dependence on records with fields, is local. Local representation is common also in the nervous system. Primary sensory–motor functions and basic emotions are wired in genetically so that a nerve impulse in a specific location has a specific meaning; it causes a certain muscle fiber to contract, for example. However, the further away from the periphery a neural circuit is, the less specific its individual neurons are. Our present understanding is that higher mental functions such as abstract thought and language depend heavily on distributed representations.

This paper demonstrates holistic distributed representation with the binary Spatter Code (Kanerva, 1994, 1995, 1996), which is a particularly simple form of Holographic Reduced Representation (HRR; Plate, 1994). The HRRs studied by Plate use real and complex vectors, but all three—real, complex, and binary—are related to each other mathematically. Plate’s thesis includes a comprehensive survey of the work leading to HRR and is highly recommended.

Holistic representation is *uniform* in the following sense: all things—objects, properties, relations, attributes, values, composed structures, mappings between structures—are random points of an enormous vector space: its dimensionality $N > 1{,}000$. The vector components (the columns) are independent and identically distributed (i.i.d.). In the real HRR they are distributed normally with zero mean and $1/N$ variance, in the complex HRR they are distributed uniformly over the unit circle (magnitude = 1), and in the binary HRR (i.e., the spatter code) 0s and 1s are equally probable. Total absence of fields sets holistic representation apart from traditional representation.

Holistic representation makes it possible to combine structure and content, or syntax and semantics, so that the distance between codewords δ, or their correlation ρ, reflects similarity of both structure and content: it reflects the similarity of *meaning*. That makes holistic representation an interesting candidate for mental representation.

Representations are synthesized from *parts* by *chunking*. A part is like a field in a record or a slot in a frame, for example, and the parts are chunked together by normalizing their sum (i.e., by superposition; thresholded sum for the binary spatter code). The chunked representation—the holistic record—correlates with its parts, as shown in Figure 4. When the dimensionality is high, even a low correlation between a chunk and its parts is significant. This matters greatly in the analysis of chunked representations. Since the chunking operator (normalized sum) is commutative, the chunked record represents the *set* of its parts.

Both single codewords and codewords combined by *binding* can serve as elements or parts for chunking. In the examples of this paper, some parts have been formed by binding a field name and a field value to each other (e.g., **name**⊗**Pat**), others by binding analogous values to each other (e.g., **Pat**⊗**Lee** in Solution 2u), and also a random vector **R** has been used as a part. The binary spatter code binds with bitwise XOR, the real HRR binds with circular convolution, and the complex HRR with coordinatewise complex multiplication (adding together of phase angles).

Chunked codewords are analyzed by *probing* and *clean-up*. A codeword that has been used for binding (e.g., **name**) is used also for probing, and the probing operator is the inverse of the binding operator. The probing operator for the binary spatter code is the same XOR as used in binding (cf. *Decoding* above), for the real HRR it is inverse convolution (also called 'correlation' and realized by convolution with a "reflected" probe), and for the complex HRR it is complex division (realized by coordinatewise multiplication with the probe's complex conjugates). It is important mathematically that the probing operator distributes over the chunking operator (with the real HRR it distributes only approximately, but that is good enough).

Probing of chunked codewords produces noisy results, and the noise increases with the number of chunked parts, *K*, according to Figure 4. However, the statistics of long, random codewords are such that the valid codeword closest to a noisy result is correct with high probability. Therefore a system based on holistic representation needs a *clean-up memory* that realizes the nearest-neighbor method. The clean-up memory could be a table of valid codewords and a procedure for finding the best-matching table entry for any noisy codeword, or it could be an autoassociative neural net that stores valid codewords as point attractors. For the neural net to work well, however, it must have large and regular basins of attraction, because the amount of noise that needs cleaning up is easily 35% (see Fig. 4). However, there is a limit to how much noise can be cleaned up reliably, which in turn limits the number of parts that can be chunked at once into a holistic record if such a record is to be analyzable.

Correlation Between Codewords. In the examples of this paper we have assumed a base set of random codewords **name**, **Pat**, **Lee**, **sex**, …, **66** that are uncorrelated. This idealization is useful in getting the discussion started, but whichever way we start, we soon deal with correlated codewords, because the idea in holistic representation is that similar meaning should be reflected in the similarity of bit patterns. For example, we would expect the codewords for similar names to be similar; **Pat** and **Pam** should correlate quite highly.

The Scatter Code of Smith and Stanford (1990) demonstrates this principle beautifully. Numbers are encoded in random *N*-bit strings so that the Hamming distance between them grows stochastically with the difference between the numbers. The codewords for adjacent integers are only a few bits apart, but as we move from one integer to the next, a few bits are chosen at random and flipped. The expected correlation between codewords decreases as a negative exponential of the difference between the numbers, and the codewords for very different integers are approximately *N*/2 bits apart. In the multidimensional Scatter Code such codewords for the individual dimensions are bound together with coordinatewise XOR (Stanford & Smith, 1994).

As new codewords are built from existing ones, they, too, are random but not necessarily uncorrelated. Codewords that share parts or that are made of similar parts are correlated. For example, if we encode the second record of Figure 2 using an uncorrelated codeword **female**, the resulting $\mathbf{PSmith}_F$ correlates with the first **PSmith** (δ = 0.25, ρ = 0.5; the conventional records differ in only one bit out of 68, and for them ρ = 0.97). So why don't added levels of composition eventually drive everything into some small part of the code space? Chunking has the tendency to do exactly that, but fortunately binding (with XOR) has the opposite tendency to scatter things about. It is significant mathematically that the binding and chunking operators complement each other in this way.

Beyond Correlation. Correlation alone apparently lacks the power to fully describe structure, such as found in language. In language research, at least, statistical approaches are only a part of the picture, and logical and symbolic approaches are often the major part. That is particularly true as regards grammar.

Holistic representation allows mapping between codewords that may meet this added need for structural description. A high correlation between codewords means that they are related—the codewords occupy some small part of the code space. But significant relations between structures—and codewords—can also exist when the codewords are uncorrelated; when they are in totally different parts of the code space. Then the relation is *analogical*. Holistic records for (Pat, male, 66) and (Lee, female, 33) would be uncorrelated, whereas the structures clearly are related. The same three variables simply have very different values. The relation between such records is by *holistic mapping*.

We already have two examples of holistic mapping. The mapping **T** in Solution 2u is composed of analogous parts of **PM6** and **LF6** by binding and chunking—the same operations that build holistic records. In Solution 3a, ($\mathbf{33}\otimes\mathbf{PF3}$)$\otimes\mathbf{LF6}$ can be interpreted as a request to find in **LF6** the part that corresponds to the **33** of **PF3**, in other words, the analogous part. It is to be noted that these mappings themselves are points of the common code space, so that it is even possible to represent and study analogies between analogies within this system.

How are holistic representations brainlike beyond how artificial neural nets at large are? They work only with large patterns (N > 1,000) that are fundamentally random, and information is distributed in the extreme, including the information about the structure of information (there are no fields). Systems scale readily to brain-size, yet composition should be done in relatively small chunks (evidence for Miller's 7 ± 2). Brains are instruments of learning; encoding new structure with holistic representation is simple, and the mapping of codewords provides a potentially useful model of analogy.

In this paper we have described holistic representation within the framework of traditional representation. However, producing the functions of traditional representation is not the final goal although it is an important criterion and a measure of credibility. The ultimate challenge and goal is to capture and describe structure in a stream of data by binding, chunking, and holistic mapping—and by whatever other useful operations are found. An added desideratum is that it should be relatively easy for us to interpret the resulting structure symbolically or logically if we so wish.

References

Anderson, J.A. (1995) *An Introduction to Neural Networks.* Cambridge, Mass.: MIT Press.

Hinton, G.E. (1990) Mapping part–whole hierarchies into connectionist networks. *Artificial Intelligence* 46(1–2):47–75.

Hinton, G.E., McClelland, J.L., and Rumelhart, D.E. (1986) Distributed representation. In D.E. Rumelhart and J.L. McClelland (eds.) *Parallel Distributed Processing,* vol. 1; 77–109. Cambridge, Mass.: MIT Press.

Kanerva, P. (1994) The Spatter Code for encoding concepts at many levels. In M. Marinaro and P.G. Morasso (eds.), *ICANN '94, Proceedings International Conference on Artificial Neural Networks*, vol. 1; 226–229. London: Springer–Verlag.

Kanerva, P. (1995) A family of binary spatter codes. In F. Fogelman-Soulie and P. Gallineri (eds.), *ICANN '95, Proceedings International Conference on Artificial Neural Networks*, vol. 1; 517–522. Paris: EC2 & Cie.

Kanerva, P. (1996) Binary spatter-coding of ordered *K*-tuples. In C. von der Malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff (eds.), *Artificial Neural Networks—ICANN 96 Proceedings*; 869–873. Berlin: Springer.

Plate, T.A. (1984) Distributed Representations and Nested Compositional Structure. PhD thesis. Graduate Department of Computer Science, University of Toronto. (Available on Internet Ftp-host: ftp.cs.utoronto.ca as Ftp-file: /pub/tap/plate.thesis.ps.Z)

Pollack, J.P. (1990) Recursive distributed representations. *Artificial Intelligence* 46(1–2):77–105.

Smith, D., and Stanford, P. (1990) A random walk in Hamming space. *Proceedings 1990 International Joint Conference on Neural Networks (IJCNN 90),* vol. 2; 465–470.

Smolensky, P. (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence* 46(1–2):159–216.

Stanford, P., and Smith, D. (1994) The multidimensional Scatter Code: A data fusion technique with exponential capacity. In M. Marinaro and P.G. Morasso (eds.), *ICANN '94, Proceedings International Conference on Artificial Neural Networks*, vol 2; 1432–1435. London: Springer–Verlag.

Touretzky, D.S. (1990) BoltzCONS: Dynamic symbolic structures in connectionist networks. *Artificial Intelligence* 46(1–2):5–46.

Touretzky, D.S., and Geva, S. (1987) A distributed connectionist representation for concept structures. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*; 155–164. Hillsdale, NJ: Erlbaum.