Analogy retrieval and processing with distributed vector representations

Tony A. Plate

Bios Group, LP, 317 Paseo de Peralta, Santa Fe, NM, 87501, USA
E-mail: tplate@attglobal.net

Abstract: *Holographic Reduced Representations (HRRs) are a method for encoding nested relational structures in fixed width vector representations. HRRs encode relational structures as vector representations in such a way that the superficial similarity of the vectors reflects both superficial and structural similarity of the relational structures. HRRs also support a number of operations that could be very useful in psychological models of human analogy processing: fast estimation of superficial and structural similarity via a vector dot-sin.25Ln fnodingcorureponodingobjlects intwol structure;l andchunkdin-*

usefu
ural neworks,h distributed representation, bfnodin,n analog,n analoy retrieva,l structur,-

bol Pprocessin,.17:1,sp@gesn29Ð40, 2000.t

1
Appearted inÒExupet Syestis:.Tth IenxrnancionalJouronal ofKnowledgs-
-wol inciostrastto feactur- vector

2 Human Analogy Processing

Analog retrieval and mapping have received a significant amount of attention in the psychological literature. Much attention has been devoted to teasing apart the differing effects of superficial and structural similarity in retrieval and mapping. A very brief introduction to the concepts and task involved in analogy processing is given here; for more details, see Forbus *et al* (1994), Thagard *et al* (1990), or Hummel and Holyoak (1997).

Analog processing concerns the processes of retrieving stories or scenarios from long-term memory in response to some immediate stimulus (another story or scenario), and relating or applying the retrieved story or scenario to the current one. In a typical experiment to test how human subjects retrieve and process analogs, a subject might be presented with a number of stories in the first session, and asked to reason about them, or merely to remember them for the future. In the second session, several days or weeks later, the same subject would be presented with a new story and asked which of the previous stories they are reminded of. They might also be asked to rate how similar the retrieved stories are to the new one, or to point out which objects in a retrieved story correspond to which objects in the new story. The new story might also involve a problem for which the retrieved story can provide a valuable structural hint about the solution method (this is an analog inference task). The types of stories or scenarios that have been used as stimuli for human experiments are typically specially written ones of the order of one paragraph in length, involving up to a dozen entities, and having a number of relationships among them. For experiments with computers, researchers have used plot-synopsis of Shakespeare's plays, Aesop's fables, and the same materials as in human experiments, encoded into some kind of predicate-calculus representation.

There are four broad categories of tasks involved in analogy processing:

- Retrieval or reminding: the process of accessing potential analogs of some probe story from long-term memory. This process is regarded as not being under conscious control – subjects are merely asked to write down or indicate all the previous stimuli that the current stimulus reminds them of.
- Judgment: rating the similarity of one story to another, after conscious consideration. Human subjects are heavily influenced by structural correspondences in making judgments of similarity.
- Mapping: finding the corresponding objects (i.e., the objects that fill the same structural roles) in two stories that share a large degree of structure.
- Inference: proposing a novel solution or consequence in one story, based on the presence of a similar solution or consequence in the other story.

In this paper, I will be concerned with the retrieval and mapping tasks, and showing how human performance on these tasks can be modeled using a simple vector-based computations.

For illustrations, the following series of episodes are used in this paper. Together, the episodes involve dogs (Fido, Spot and Rover), people (Jane, John and Fred), a cat (Felix) and a mouse (Mort). Members of one species are assumed to be similar to each other but not to members of other species. The "probe" episode (denoted "**P**") to which the others are compared, is "Spot bit Jane, causing Jane to flee from Spot". There are five other episodes, which have different combinations of types of similarity to the probe (all share predicates with the probe):

- **LS** (Literal Similarity) "Fido bit John, causing John to flee from Fido." (Has both structural and superficial similarity to the probe **P**.)
- **SF** (Surface features) "John fled from Fido, causing Fido to bite John." (Has superficial but not structural similarity.)
- **CM** (Cross-mapped analogy) "Fred bit Rover, causing Rover to flee from Fred." (Has both structural and superficial similarity, but types of corresponding objects are switched.)
- **AN** (Analogy) "Mort bit Felix, causing Felix to flee from Mort." (Has structural but not superficial similarity).
- **FOR** (First-order-relations only) "Mort fled from Felix, causing Felix to bite Mort." (Has neither structural nor superficial similarity, other than shared predicates.)

It is generally accepted that in adults, structural similarity plays a large role in analogical mapping and conscious similarity judgements. The role of structural similarity in retrieval is less clear: some researchers argue that structural similarity usually has little effect on retrieval (Gentner, Rattermann, and

Forbus 1993) while others argue that under some circumstances, structural similarity can influence retrieval (Wharton, Holyoak, Downing, Lange, Wickens, and Melz 1994). Others suggest that structural similarity matters only when the entities involved in the situations share superficial features (Ross 1989). There is strong evidence that cross-mapping results in lower retrieval performance compared to the literal similarity condition (Ross 1989). Overall, the general consensus is that the pattern for retrievability of items from long-term memory seems to be LS > CM ≥ SF > AN ≥ FOR (the disputes are to do with whether the inequalities are strict or not, or the conditions under which they are strict).

The primary result concerning human performance on mapping is that people, especially children, are far more prone to making erroroneous mappings in situations where cross mappings are present, e.g., as in the cross-mapped episode above, where the correct, class-crossing mappings are “dog to person” and “person to dog”, while alternate, but incorrect, class-preserving mappings of “dog to dog” and “person to person” are available (Ross 1989, Gentner et al, 1993).

Existing computational models of human performance on analog retrieval tasks such as ARCS (Thagard, Holyoak, Nelson and Gochfeld, 1990), and MAC/FAC (Forbus, Gentner and Law, 1994) have explained the encoding retrieval data by invoking two processes. The first is a simple one based on superficial similarities. This explains much of the human performance, but cannot account for effects of structural similarity (i.e., LS > SF, and AN ≥ FOR). Thus, these models require a second process that takes structural similarity into account, which involves additional complex computation. In this paper I will argue that it is possible to explain the pattern of retrieval ability observed in people with a single-stage model based on vector matching of HRRs. Comparisons in this model will be primarily sensitive to superficial similarity, but will also be sensitive to structural similarity, thus mirroring the retrieval preferences of people; LS > SF, and AN ≥ FOR (the latter being greater or equal, depending on details). With HRRs it is also possible to make estimates of corresponding objects that are usually correct, except for cross-mappings, again mirroring human performance.

3 Vector Representations

There are two types of vector representation: localist and distributed. Structure can be encoded in localist vector representations, but size of the vectors becomes unreasonably large when realistic numbers of features are included. Encoding structure in distributed representations requires binding operation. This section describes circular convolution and how it can be used as a binding operation to encode structural information in distributed representations. The resulting scheme is called Holographic Reduced Representations.

3.1 Vector operations

The two vector operations commonly used with vector representations are superposition (i.e., addition) and similarity (i.e., dot-product or cosine). These two vector operations, and other scalar-vector operations such as scaling and normalization, are sufficient for interesting and useful memory models. They allow one to represent items as sets (superpositions) of features and compute the similarity of items, as in the feature summation models of McClelland and Rumelhart (1981) and Anderson (1983).

HRRs use circular convolution for binding operations. Circular convolution has

Using the three-operation vocabulary of HRRs (superposition, convolution, and similarity), nested compositional structure can be represented in vectors in a practical and elegant manner. HRRs have several high-level properties that distinguish them from non-structural vector representations and make them interesting for use in psychological memory models. One is that they provide a natural and systematic method for chunking and representing the relationships among chunks. Another is that some aspects of structure are expressed as surface features in HRRs, which means that vector dot-products of HRRs can provide indications of structural similarity. Yet another is that some "higher-level" operations such as guessing at corresponding objects can be done quickly.

3.2 From local to distributed representations

Vector representations come in two flavors: local and distributed. Each involves different tradeoffs of the properties of the representation space. For models involving only a small number of features and a few simple objects, local representations can be more parsimonious. Because of their greater representational efficiency, distributed representations come into their own for models in which there are large numbers of objects and features.

In local vector representations each vector element or unit indicates the strength, or merely presence or absence, of a particular feature in an object, or an object in a situation. For example, in the vector representation used in the MAC stage of MAC/FAC (Forbus, *et al* 1994), a sparse, high-dimensional "content vector" is used to summarize each story (an item). Each vector element is a count of the number of times a particular predicate or attribute occurs in the story.

In distributed representations features of items are represented by patterns in the vector elements rather than single elements (Hinton, McClelland and Rumelhart, 1986). Each vector element participates in the representation of many features, and each feature is represented collectively by many vector elements. Under not-particularly restrictive conditions, several patterns can be superimposed without losing the identity of the component patterns. Like the localist content vector, the distributed version can encode the presence of several features.

In some respects, localist and distributed representations are equivalent. They can have almost indistinguishable forms when features are numerous and fine-grained. Also, local representations can be mapped to distributed ones by a simple linear map, and back by a thresholded linear map. However, there is one difference very pertinent to the representation of structure, which concerns scaling of numbers of features with vector dimensionality. This difference is that the total number of possible features is limited to the number of elements in the vector for localist representations, but is exponential in vector dimensionality for distributed representations. Users of localist feature vectors have recognized this problem: Forbus, Gentner and Law (1994) acknowledge that the content vectors used in their MAC stage will have problems scaling to psychologically plausible representations containing hundreds of thousands of predicates. This scaling problem becomes of critical importance when we consider using local feature vectors to represent structural information. The content-vector style of representation does not capture structure — "Spot bit Jane" has exactly the same content vector as "Jane bit Spot." Wharton *et al* (1994) point out that localist vector representations could be augmented with combinatorial features to represent all possible structural combinations, but note that when applied recursively to represent things such as the sour-grapes feature "thing that is desired but can't be obtained and hence is denigrated" this would result in a combinatorial explosion in the size of vectors.

This restrictive limitation on the number of features does not apply to distributed representations (unless for some reason patterns are required to be perfectly orthogonal, which is often not necessary). The limitation is escaped by taking advantage of the property of high-dimensional vector spaces that the number of approximately-independent directions or patterns (specified as a tolerance angle) grows exponentially with the number of dimensions (Plate 1994). Each of these approximately-independent patterns can represent a feature, and thus hundreds of thousands of "virtual" features can be represented in a vector with far fewer elements. This involves two-tradeoffs. The first is that only relatively few features can be present at once. However, this is usually the case when there are large numbers of potential features. The second is that there is a chance of interference (ghosting or cross-talk) when too many features are present at once. The probability of interference can be made arbitrarily low by limiting the number of features present at once and choosing a large enough dimension of the vectors.

Given that distributed representations provide a sufficient number "virtual" features for the representation of combinatorial features, what is needed is a systematic way of generating and decoding the patterns which represent combinatorial features. This is the role of the binding operation. As a binding operation, circular convolution provides a fast, systematic, and reversible way of constructing new patterns to represent combinatorial features. The remainder of this section explains circular convolution and the encoding and decoding of HRRs.

3.3 Associations & complex structure

The first step to representing nested relational structure is to represent flat relational structure. This can be done with associations, or bindings, between roles and fillers. Suppose we have a predicate representation for "Spot bit Jane": bite(spot,jane). Spot is the agent of this relation, and Jane is the object (or patient). A distributed representation of this relation must be careful to preserve the information about which person is associated with which role (agent or object) so that there is no confusion with "Jane bit Spot".

One obvious way to avoid ambiguity about what is associated with which role is to divide the vector into several blocks, and devote a block to each role (Hinton, 1990). The drawback of this method is that it is unsuitable for representing recursively nested relations such as "Spot bit Jane, causing Jane to flee from Spot": cause(bite(spot,jane),flee(jane,spot)). An alternative is to use a function that binds roles and fillers by creating a new pattern from the role and filler patterns, and represent a relation as the superposition of role-filler binding patterns. This is the approach taken by Smolensky (1990) with tensor products and Pollack (1990) with RAAMs, and is also the approach taken with HRRs, though in each case the function used to create a new pattern is different.

In order to represent nested relational structure, it is necessary to be able to apply the binding operation recursively in order to form bindings between role patterns and patterns representing entire relations, e.g., between the cause-antecedent role and the pattern representing bite(spot,jane). HRRs use circular convolution as the binding operation. Circular convolution can be viewed as an associative memory operator that produces noisy but compact representations of associations. Various types of convolution have been used as the associative operators in a number of computational and psychological associative memory models (Willshaw, Buneman, and Longuet Higgins, 1969, Borsellino and Poggio 1973, Liepa 1977, Willshaw 1981, Murdock 1982, Metcalfe 1982, Murdock 1987, Paek and Psaltis 1987, Plate 1994, 1995). Circular convolution is particularly suited to recursive application, and thus the representation of hierarchical structure, because the number of elements in a binding is the same as in the role and filler.

3.4 Circular convolution

Circular convolution maps two real-valued n-dimensional vectors onto one. If $\mathbf{x}$ and $\mathbf{y}$ are n-dimensional vectors (subscripted 0 to n-1), then the elements of $\mathbf{z} = \mathbf{x} \circledast \mathbf{y}$ are

$$z_i = \sum_{k=0}^{n-1} x_k y_{i-k}$$

where subscripts are taken modulo-n. and $\circledast$ denotes circular convolution. Circular convolution can be viewed as a compression of the outer (or tensor) product of the two vectors, as shown in Figure 1. Each of the small circles represents an element of the outer product of $\mathbf{x}$ and $\mathbf{y}$, e.g., the middle bottom one is x_2y_1. The elements of the circular convolution of $\mathbf{x}$ and $\mathbf{x}$ are the sums of the outer product elements along the wrapped diagonal lines.

<Figure 1 about here>

Circular convolution can be regarded as a multiplication operator for vectors and has many algebraic properties in common with scalar and matrix multiplication. It is commutative ($\mathbf{x} \circledast \mathbf{y}=\mathbf{y} \circledast \mathbf{x}$), associative ($\mathbf{x} \circledast (\mathbf{y} \circledast \mathbf{z})=(\mathbf{x} \circledast \mathbf{y}) \circledast \mathbf{z}$), and bilinear ($\mathbf{x} \circledast (\alpha\mathbf{y}+\beta\mathbf{z})=\alpha\mathbf{x} \circledast \mathbf{z}+\beta\mathbf{x} \circledast \mathbf{z}$). There is an identity vector $\mathbf{I}$ ($\mathbf{I} \circledast \mathbf{x}=\mathbf{x}$) and a zero vector $\bar{0}$ ($\bar{0} \circledast \mathbf{x}=\bar{0}$). Inverses $\mathbf{x}^{-1}$ exist for most vectors ($\mathbf{x}^{-1} \circledast \mathbf{x}=\mathbf{I}$).

An association between two items $\mathbf{x}$ and $\mathbf{y}$ can be represented by the convolution of the two items: $\mathbf{x} \circledast \mathbf{y}$. The inverse vector of $\mathbf{x}$ can be used to reconstruct $\mathbf{y}$ from $\mathbf{x} \circledast \mathbf{y}$: $\mathbf{x}^{-1} \circledast (\mathbf{x} \circledast \mathbf{y})=\mathbf{y}$. However, except under certain restrictive conditions, the inverse is numerically unstable and is not always the best choice

for decoding. For vectors which have randomly chosen elements independently distributed as N(0,1/n) (the normal distribution with mean 0 and variance 1/n) there is an approximate inverse with attractive properties. The approximate inverse of $\mathbf{x}$ is denoted by $\mathbf{x}^T$. It is a simple rearrangement of the elements of $\mathbf{x}$: $x^T_i = x_{-i}$, where subscripts are modulo-n. The approximate inverse is simple to compute and is numerically stable. Reconstruction using the approximation inverse is noisy. The convolution product $\mathbf{x}^T \ \mathbf{x} \ \mathbf{y}$ can be written as $\mathbf{y}+\eta$, where the η can be considered as zero-mean noise whose magnitude (variance) decreases with increasing vector dimension.

Multiple associations can be represented by the sum of the individual associations. For example, suppose $\mathbf{x}$, $\mathbf{y}$, $\mathbf{v}$, and $\mathbf{w}$ are all randomly chosen vectors with elements independently distributed as N(0,1/n). The association of $\mathbf{x}$ with $\mathbf{y}$ and $\mathbf{v}$ with $\mathbf{w}$ can be represented by $\mathbf{z}=\mathbf{x} \ \mathbf{y}+\mathbf{v} \ \mathbf{w}$. To find what is associated with $\mathbf{x}$ we convolve $\mathbf{z}$ with $\mathbf{x}^T$. The result can be expressed as $\mathbf{x}^T \ \mathbf{x} \ \mathbf{y}+\mathbf{x}^T \ \mathbf{v} \ \mathbf{w}$. The first term here is approximately equal to $\mathbf{y}$, since $\mathbf{x}^T \ \mathbf{x} \approx \mathbf{I}$. The second term can be regarded as noise – it will not be highly correlated with any of $\mathbf{x}$, $\mathbf{y}$, $\mathbf{v}$, or $\mathbf{w}$, or any other randomly chosen vector, provided that the vector dimensionality is large. The sum of the two terms will be recognizable as a distorted version of $\mathbf{y}$. The vector dimension required for this to work well with a moderate number of associations is quite high – it must be in the hundreds to thousands. For further discussion and quantitative analysis, see Plate 1994.

3.5 Similarity preservation and randomization

Convolution and superposition have complementary properties with respect to how they preserve and destroy similarity of patterns.

Convolution preserves both similarity and lack of similarity in a multiplicative fashion: the similarity of two role-filler binding patterns is approximately equal to the product of the similarities of the respective role and filler patterns (provided that the role patterns are not similar to the filler patterns.) Thus, if two bindings have the same role, their similarity will be equal to that of the fillers. Conversely, if two roles have no similarity, bindings involving them will have similarity regardless of the fillers. Furthermore, convolution is randomizing in that role-filler binding patterns are not similar to either the role or filler patterns.

Superposition behaves differently: a superposition of patterns remains similar to each individual pattern, to a decreasing degree as more patterns are superimposed. This can be seen as an unstructured and additive kind of similarity preservation.

3.6 HRRs for relational structure

Consider representing a nested proposition such as “Spot bit Jane, causing Jane to flee from Spot” in a vector pattern. We would like this pattern to faithfully record structure and also to be suitable for detecting at least superficial similarity by computing dot-products.

The structure of a proposition can be represented by superimposing patterns representing the predicate name and the role-filler bindings. This provides a structural skeleton that faithfully records structure.

The skeleton HRR for the proposition “Spot bit Jane” is constructed as follows:

$$\mathbf{K_{P\text{-}bite}} = \mathbf{bite} + \mathbf{bite}_{agt} \ \mathbf{spot} + \mathbf{bite}_{obj} \ \mathbf{jane}$$

The pattern **bite** represents the predicate label, $\mathbf{bite}_{agt}$ and $\mathbf{bite}_{obj}$ its roles, and **spot** and **jane** the entities “Spot” and “Jane”. If we have the pattern $\mathbf{K_{P\text{-}bite}}$ and know the role patterns, then we can reconstruct the filler patterns by convolving $\mathbf{K_{P\text{-}bite}}$ with the approximate inverses of the role patterns. For example, $\mathbf{bite}_{agt}^T \ \mathbf{K_{P\text{-}bite}}$ gives a noisy version of **spot** which, if necessary, can be cleaned up using an auto-associative item memory. The pattern **bite** is made a component of $\mathbf{K_{P\text{-}bite}}$ in order to identify it as a *bite* proposition and thus allow a system to deduce that the appropriate role patterns for decoding are $\mathbf{bite}_{agt}$ and $\mathbf{bite}_{obj}$.

The skeleton HRR pattern for the proposition “Spot bit Jane” is an n-dimensional pattern just like the patterns **spot**, **bite**, etc. Thus, it is easily used as a filler in a higher-order proposition. For example, the skeleton HRR $\mathbf{K_P}$ representing “Spot bit Jane, which caused Jane to flee from Spot” is constructed as follows:

$$\mathbf{K_{P\text{-}flee}} = \mathbf{flee} + \mathbf{flee}_{agt} \ \mathbf{jane} + \mathbf{flee}_{from} \ \mathbf{spot}$$

$$\mathbf{K_P} = \mathbf{cause} + \mathbf{cause}_{antc} \ \mathbf{K_{P\text{-}bite}} + \mathbf{cause}_{cnsq} \ \mathbf{K_{P\text{-}flee}}$$

Such higher-level HRRs can be decoded in the same way as first order HRRs. For example, the propositional filler of the cause antecedent role is decoded as follows:

$$\mathbf{K_P}\ \ \mathbf{cause}_{antc}{}^{T} \approx \mathbf{K_{P\text{-}bite}}$$

This could be cleaned up and then decoded again to discover its fillers. Alternatively, lower-level fillers can be decoded more quickly but less reliably with no intermediate cleanup:

$$\mathbf{K_P}\ \ \mathbf{cause}_{antc}{}^{T}\ \ \mathbf{bite}_{agt}{}^{T} \approx \mathbf{spot}$$

The result of such fast decoding might be so noisy that it lacks sufficient detail to be recognizable as **spot**, but, providing that all dogs are represented by similar patterns, it could still be recognizable as a dog.

The other goal for a vector representation was that patterns should reflect superficial similarity, i.e., two patterns should be similar if the structures they represent merely involve similar fillers or predicates. The presence of predicate labels in HRRs ensures that patterns for the same predicate are similar. However, skeleton HRRs do not behave as desired with respect to fillers: the randomizing properties of convolution mean that $\mathbf{role_1}\ \ \mathbf{filler_1}$ is only similar to $\mathbf{role_2}\ \ \mathbf{filler_2}$ to the extent that $\mathbf{role_1}$ is similar to $\mathbf{role_2}$ *and* $\mathbf{filler_1}$ is similar to $\mathbf{filler_2}$. However, HRRs are easily made to reflect superficial similarity by superimposing the filler patterns together with the structural skeleton HRR. Thus, the fleshed-out HRR for "Spot bit Jane" is as follows:

$$\mathbf{P_{bite}} = \mathbf{bite} + \mathbf{spot} + \mathbf{jane} + \mathbf{bite}_{agt}\ \ \mathbf{spot} + \mathbf{bite}_{obj}\ \ \mathbf{jane}$$

Adding in the fillers makes decoding more noisy, but does not prevent successful decoding. For higher level propositions, the same idea of adding in fillers can be applied recursively. For example, the HRR for "Spot bit Jane, causing Jane to flee from Spot" is constructed as follows:

$$\mathbf{P_{flee}} = \mathbf{flee} + \mathbf{spot} + \mathbf{jane} + \mathbf{flee}_{agt}\ \ \mathbf{jane} + \mathbf{flee}_{from}\ \ \mathbf{spot}$$

$$\mathbf{P} = \mathbf{cause} + \mathbf{P_{bite}} + \mathbf{P_{flee}} + \mathbf{cause}_{antc}\ \ \mathbf{P_{bite}} + \mathbf{cause}_{cnsq}\ \ \mathbf{P_{flee}}$$

HRRs constructed like this will be similar if they merely involve similar entities or predicates. Because of the similarity preserving properties of convolution, they will be even more similar if the entities are involved in similar roles. Thus it turns out that the similarity of HRRs can reflect both superficial and structural similarity in a way which neatly corresponds to the data on human analog retrieval.

3.7 Need for a "clean-up" memory

Convolution encodings are remarkably compact: a number of associations between *n*-dimensional patterns packed into one *n*-dimensional pattern. The price we pay for this compactness is noise in decoded vectors. Consequently, if we want a convolution-based associative-memory model to provide accurate reconstructions of decoded patterns, it must be equipped with an additional error-correcting autoassociative item memory. This can clean up the noisy patterns retrieved from the convolution encodings. This clean-up memory must store all the items that the system can produce. When given a noisy version of one of those items it must either output the closest item or indicate that the input is not close to any of the stored items. Note that only a few associations are stored as convolution encodings in a single pattern, whereas many patterns are stored in the clean-up memory.

The clean-up memory can be implemented in any number of ways, as discussed in Section 5. All that is required of it is that when it is presented with a pattern, it should retrieve the closest stored pattern. For the experiments described in this paper, the item memory was implemented as a list of vectors with access by exhaustive comparison.

In order to support faithful decoding, the clean-up memory must contain all patterns that are potential targets of decoding. This includes patterns for the following that have been previously encountered and are components in a higher-level pattern in the clean-up memory: (a) objects, such as **spot** and **jane**; (b) roles, such as $\mathbf{bite}_{obj}$ and $\mathbf{cause}_{antc}$; (c) predicate labels, such as **bite** and **cause**; and (d) propositions of all levels, such as $\mathbf{P_{bite}}$ and **P** (i.e., the top-level propositions and their sub-propositions – the "chunks" discussed in Section 6). The clean-up memory does not need to contain individual role-filler bindings, such as $\mathbf{bite}_{agt}\ \ \mathbf{spot}$, unless the model requires these as potential targets of decoding. Nor does the clean-up memory need to contain propositions that have not been previously encountered, or are not a component of some higher level item, as such patterns are not potential targets of decoding.

3.8 Normalization

The final point to consider when constructing HRRs is maintaining the overall strength of patterns and the statistical distribution of their elements. The easiest way to do this is to normalize all patterns to have a Euclidean length of one. Here, the normalized version of the vector **x** is denoted by $\langle \mathbf{x} \rangle$ and is defined as follows (the denominator is the Euclidean length (the magnitude) of **x**):

$$\langle \mathbf{x} \rangle = \mathbf{x} \,/\, \sqrt{\sum_{i=0}^{n-1} x_i^2}$$

There are several side-benefits to normalization. Firstly, there is no concern over the magnitudes of patterns affecting dot-product similarity scores. Secondly, when two normalized patterns are superimposed each has equal weight in the result, though it is still possible to weight components to cause one to be more prominent in the sum. Thirdly, normalization reduces the noise in recognizing decoded fillers (Plate, 1994).

3.9 Classes and instances

In many connectionist systems, the patterns representing items are analyzed in terms of "micro-features". Each element of the vector is considered to represent a simple feature. For example, Hinton's (1990) family trees network learned features representing concepts such as age and nationality. The requirement of HRRs that elements of vectors be randomly and independently distributed seems at odds with this interpretation.

However, it is not necessary to have a one-to-one correspondence between features (or micro-features) and vector elements. Features can also be represented by patterns. An item having some features can be partly the sum of those features. Furthermore, while all the instances in a class might share a set of features, instances can be distinguished from each by the addition of some identity-denoting random vector that is unique for each instance. For example, "John" can be represented by **john**=**animal**+**human**+$\mathbf{id}_{\text{john}}$, where **animal** and **human** are components of all instances of those classes and $\mathbf{id}_{\text{john}}$ is a random vector that distinguishes **john** from patterns for other humans. Importance or salience of component features can be adjusted by weighting. This scheme also allows for instances of subclasses to inherit the features of superclasses.

4 Estimating Similarity

The six "dog bites human" episodes provide a simple demonstration that HRR scores can reflect similarity of structural arrangements, as well as similarity of surface features. It also and demonstrates that a model based on HRR similarity scores alone can explain the pattern of retrieval observed in people: LS > CM ≥ SF > AN ≥ FOR.

The HRRs for the probe (**P**) and the literally similar episode ($\mathbf{E_{LS}}$) are constructed as follows:

$\mathbf{P_{bite}} = \mathbf{bite} + \mathbf{spot} * \mathbf{jane} + \mathbf{bite}_{\text{agt}} * \mathbf{spot} + \mathbf{bite}_{\text{obj}} * \mathbf{jane}$

$\mathbf{P_{flee}} = \mathbf{flee} + \mathbf{spot} * \mathbf{jane} + \mathbf{flee}_{\text{agt}} * \mathbf{jane} + \mathbf{flee}_{\text{from}} * \mathbf{spot}$

$\mathbf{P} = \mathbf{cause} + \mathbf{P_{bite}} * \mathbf{P_{flee}} + \mathbf{cause}_{\text{antc}} * \mathbf{P_{bite}} + \mathbf{cause}_{\text{cnsq}} * \mathbf{P_{flee}}$

$\mathbf{E_{LS\text{-}bite}} = \mathbf{bite} + \mathbf{fido} * \mathbf{john} + \mathbf{bite}_{\text{agt}} * \mathbf{fido} + \mathbf{bite}_{\text{obj}} * \mathbf{john}$

$\mathbf{E_{LS\text{-}flee}} = \mathbf{flee} + \mathbf{fido} * \mathbf{john} + \mathbf{flee}_{\text{agt}} * \mathbf{john} + \mathbf{flee}_{\text{from}} * \mathbf{fido}$

$\mathbf{E_{LS}} = \mathbf{cause} + \mathbf{E_{LS\text{-}bite}} * \mathbf{E_{LS\text{-}flee}} + \mathbf{cause}_{\text{antc}} * \mathbf{E_{LS\text{-}flee}} + \mathbf{cause}_{\text{cnsq}} * \mathbf{E_{LS\text{-}bite}}$

The HRRs for the other episodes are built in an analogous fashion. The patterns for instances of the same class are designed to be similar. Thus Jane, John, and Fred, being people, are represented by similar patterns, as are the dogs Spot, Fido, and Rover. The different species, people, dogs, cats and mice, are not considered similar at all — the examples are simple enough not to require a hierarchy of classes. The complete set of base vectors and instances used in this experiment is shown in Table 1. All base and identity (**id**) vectors were randomly chosen with elements independently distributed as N(0,1/*n*).

Base vectors		Instance vectors
person	**bite**	**jane** = **person** + $\mathbf{id}_{\text{jane}}$
dog	**flee**	**john** = **person** + $\mathbf{id}_{\text{john}}$

cat	**cause**	**fred** = **person** + **id**$_{\text{fred}}$
mouse		**spot** = **dog** + **id**$_{\text{spot}}$
		fido = **dog** + **id**$_{\text{fido}}$
bite$_{\text{agt}}$	**bite**$_{\text{obj}}$	**rover** = **dog** + **id**$_{\text{rover}}$
flee$_{\text{agt}}$	**flee**$_{\text{obj}}$	**felix** = **cat** + **id**$_{\text{felix}}$
cause$_{\text{antc}}$	**cause**$_{\text{cnsq}}$	**mort** = **mouse** + **id**$_{\text{mort}}$

Table 1

Average HRR similarity scores are shown in Table 2. These are from 100 runs with different random base and identity vectors, and a vector dimension of 2048. The directions of differences between average similarity scores were reliable – the standard deviation of the scores ranged between 0.016 and 0.026.

For comparison, MAC-style similarity scores are also shown. These are based on the dot product of normalized content vectors over the following features: the entities *person*, *dog*, *mouse*, and *cat* and the predicates *cause*, *bite*, and *flee*. For example, the content vector for the probe is $(1,1,0,0,1,1,1)/\sqrt{5}$.

The pair of episodes $\mathbf{E_{LS}}$ and $\mathbf{E_{SF}}$ each have the same surface commonalities (object features and predicate names) with the probe. The difference between them is that $\mathbf{E_{LS}}$ is structurally isomorphic to the probe, while $\mathbf{E_{SF}}$ is not. Because there is no structural information beyond the names of the higher-order predicates encoded in content vectors, $\mathbf{E_{LS}}$ and $\mathbf{E_{SF}}$ have the same content-vector similarity to the probe. On the other hand, the HRR similarity scores indicates that $\mathbf{E_{LS}}$ is more similar to the probe than $\mathbf{E_{SF}}$.

When episodes do not share object attributes with the probe, HRR scores are low and do not always reflect structural match. Although in Table 2 the HRR score for $\mathbf{E_{AN}}$ is higher than for $\mathbf{E_{FOR}}$ (due to the "bite" and "flee" propositions filling the same roles in $\mathbf{E_{AN}}$ as in the probe), this difference is not reliable across a range of AN and FOR episodes: it is possible to construct other FOR examples that have a higher score than AN examples (Plate, 1994).

$\mathbf{E_{CM}}$ is a cross-mapped analogy. It has the same structure and types of objects as the probe, but unlike $\mathbf{E_{LS}}$ and the probe, the similar objects do not map to each other (the dog maps to the person, and the person maps to the dog). Since HRR similarity scores are sensitive to having similar objects fill similar roles, $\mathbf{E_{CM}}$ has a lower HRR similarity to the probe than $\mathbf{E_{LS}}$. In contrast, the content-vector similarities of $\mathbf{E_{CM}}$ and $\mathbf{E_{LS}}$ to the probe are the same.

Overall, vector dot-products of HRRs model the observed pattern of retrievability in humans: LS > CM ≥ SF > AN ≥ FOR. They capture in a simple way the dominant effect of superficial similarity (LS & SF > AN), the positive effects of shared structure (LS > SF always, and AN > FOR sometimes, depending on details; see Plate 1994) and the negative effects of cross-mapping (LS > CM). In contrast,

		Commonalities with probe			Similarity scores	
P	Spot bit Jane, causing Jane to flee from Spot.	Object attributes	First-order relation names	Higher-order structure		
	Episodes in long-term memory:				HRR	MAC
$\mathbf{E_{LS}}$	Fido bit John, causing John to flee from Fido.	✓	✓	✓	0.71	1.0
$\mathbf{E_{SF}}$	John fled from Fido, causing Fido to bite John	✓	✓	×	0.47	1.0
$\mathbf{E_{CM}}$	Fred bit Rover, causing Rover to flee from Fred.	✓	✓	✓	0.47	1.0
$\mathbf{E_{AN}}$	Mort bit Felix, causing Felix to flee from Mort.	×	✓	✓	0.42	0.6
$\mathbf{E_{FOR}}$	Mort fled from Felix, causing Felix to bite Mort.	×	✓	×	0.30	0.6

Table 2

the vector dot-products of content vectors (the MAC model) only capture the dominant effect of superficial similarity.

4.1 Why HRR dot-products reflect structural similarity

HRR dot-products reflect structural similarity because of the presence of components representing higher-order (combinatorial) features, like $\mathbf{bite}_{agt}$ **spot**, $\mathbf{cause}_{antc}$ **bite**, and $\mathbf{cause}_{antc}$ $\mathbf{bite}_{agt}$ **spot**. These components in turn contain combinatorial features involving inherited features, like $\mathbf{bite}_{agt}$ **dog** (because **spot**= **dog**+$\mathbf{id}_{spot}$).

All of these higher-order features derive from role-filler bindings. Consequently, the HRRs described here reflect differences in structural similarity when those differences concern whether or not similar objects fill similar roles. Hence the large difference between $\mathbf{E_{CM}}$ and $\mathbf{E_{LS}}$ in their HRR similarity scores with the probe. Although $\mathbf{E_{CM}}$ and $\mathbf{E_{LS}}$ have the same component objects and isomorphic structure, $\mathbf{E_{CM}}$ does not have similar objects filling the same roles as in the probe, whereas $\mathbf{E_{LS}}$ does. Thus, $\mathbf{E_{CM}}$ has class-level combinatorial features like $\mathbf{bite}_{agt}$ **person**, which are not at all similar to those like $\mathbf{bite}_{agt}$ **dog** in the probe, while $\mathbf{E_{LS}}$ has the same class-level combinatorial features as the probe.

This pattern of sensitivity to structural similarity, in which structural similarity is only detected when similar objects fill similar roles, is very similar to the pattern observed by Ross (1989) in experiments with people. Ross found that shared structure enhanced retrieval in the presence of similar objects, provided that corresponding objects were similar, and that cross-mapping inhibited retrieval.

5 Interpretations of an Analogy

Retrieval of analogies is only the first step in many analogy processing tasks. After retrieving a potentially analogous episode we may want to decode the structure in order to evaluate more accurately the degree of structural consistency, or to use the episode for analogical reasoning. The structure of a HRR could be decoded using the techniques described in Section 2, and then used in a symbolic processor like SME or in some other connectionist architecture. However, some apparently more symbolic tasks, like finding corresponding entities, and thus deriving an interpretation of an analogy, can be computed with vector operations directly on HRRs.

Consider the probe **P** "Spot bit Jane, causing Jane to flee from Spot", and $\mathbf{E_{LS}}$ "Fido bit John, causing John to flee from Fido." The entity corresponding to Jane (which is John) can be found in two steps:

1. Extract the roles Jane fills in the probe with the operation:

$$\textbf{jane-roles} = \mathbf{P}\ \ \textbf{jane}^{T}$$

This pattern is a blend of various roles and other noise patterns. The following are the positive dot-products of the **jane-roles** pattern with other role patterns:

$$\textbf{jane-roles}\ \ \mathbf{cause}_{antc} = 0.20$$
$$\textbf{jane-rol}\text{es}\ \ \mathbf{cause}_{cnsq} = 0.18$$
$$\textbf{jane-roles}\ \ \mathbf{flee}_{agt} = 0.13$$
$$\textbf{jane-roles}\ \ \mathbf{bite}_{obj} = 0.12$$

Note that **jane** fills the roles $\mathbf{cause}_{antc}$ and $\mathbf{cause}_{cnsq}$ because **jane** is a component of $\mathbf{P_{bite}}$ and $\mathbf{P_{flee}}$, which are bound to those roles.

2. Use **jane-roles** to extract the fillers from $\mathbf{E_{LS}}$ and compare (dot-product) with the entities in $\mathbf{E_{LS}}$:

$$\mathbf{E_{LS}}\ \ \textbf{jane-roles}^{T}\ \ \textbf{john} = 0.38$$
$$\mathbf{E_{LS}}\ \ \textbf{jane-roles}^{T}\ \ \textbf{fido} = 0.05$$

The most similar pattern is **john**, which is in fact the entity in $\mathbf{E_{LS}}$ corresponding to Jane.

Table 3 shows the extraction, without intermediate role clean-up, of the entities corresponding to Jane in the various episodes. Correct extractions are checkmarked, and cases where there is no clear corresponding object have a question mark.

$\mathbf{E_{LS}}$	$\textbf{jane-roles}^{T}$	**john**	0.38	✓
		fido	0.07	

$\mathbf{E_{CM}}$ **jane-roles**^{T}	**fred**	0.25	×
	rover	0.17	
$\mathbf{E_{AN}}$ **jane-roles**^{T}	**felix**	0.16	✓
	mort	0.09	
$\mathbf{E_{SF}}$ **jane-roles**^{T}	**john**	0.23	?
	fido	0.07	
$\mathbf{E_{FOR}}$ **jane-roles**^{T}	**mort**	0.11	?
	felix	0.06	

Table 3

The correct answer is obtained in $\mathbf{E_{LS}}$, where corresponding objects are similar, and in $\mathbf{E_{AN}}$, where there is no object similarity. This extraction process has a bias towards choosing similar entities as the corresponding ones, which leads to a reasonable answer for $\mathbf{E_{SF}}$ and an incorrect answer $\mathbf{E_{CM}}$. There are no correct answers for $\mathbf{E_{SF}}$ and $\mathbf{E_{FOR}}$, because there are no consistent mapping between **P** and those episodes. However, because of the bias for mapping similar items, Fred is strongly indicated to be the entity in $\mathbf{E_{SF}}$ corresponding to Jane. The only wrong answer is given for the cross-mapped analogy $\mathbf{E_{CM}}$ where again the more similar object is indicated to be the corresponding one. Again, the effect of cross-mapping is similar to that observed by Ross (1989) in people: cross-mapping causes less accurate mapping performance.

Closer examination of the extraction process reveals both the reason for this bias and several ways of eliminating it, if that should be desired. Consider patterns containing just two of the components from **P** and $\mathbf{E_{CM}}$:

$\mathbf{P'} = \mathbf{cause} + \mathbf{bite}_{obj}\ \ \mathbf{jane}$

$\mathbf{E'_{CM}} = \mathbf{cause} + \mathbf{bite}_{obj}\ \ \mathbf{rover}$

The roles of Jane in **P′** are calculated as follows:

$\mathbf{jane\text{-}roles'} = \mathbf{P'}\ \ \mathbf{jane}^T$

$\approx \mathbf{cause}\ \ \mathbf{jane}^T + \mathbf{bite}_{obj}$

The role pattern $\mathbf{bite}_{obj}$ (and other role patterns like $\mathbf{flee}_{agt}$ and $\mathbf{cause}_{antc}$ in the full version of $\mathbf{P}\ \ \mathbf{jane}^T$) are what are wanted here. The other patterns like $\mathbf{cause}\ \ \mathbf{jane}^T$, which are not roles at all, are the source of the same-class bias in finding the corresponding object. When $\mathbf{jane\text{-}roles'}^T$ is used to extract the fillers from $\mathbf{E_{CM}}$, we get the following:

$\mathbf{corresp} = \mathbf{jane\text{-}roles'}^T\ \ \mathbf{E'_{CM}}$

$\approx (\mathbf{cause}\ \ \mathbf{jane}^T + \mathbf{bite}_{obj})^T\ \ (\mathbf{cause} + \mathbf{bite}_{obj}\ \ \mathbf{rover})$

$= \underline{\mathbf{jane}} + \mathbf{cause}\ \ \mathbf{jane}^T\ \ \mathbf{bite}_{obj}\ \ \mathbf{rover} + \mathbf{bite}_{obj}{}^T\ \ \mathbf{cause} + \underline{\mathbf{rover}}$

This includes the pattern **rover** as desired, but includes the pattern **jane** (from $(\mathbf{cause}\ \ \mathbf{jane}^T)^T\ \ \mathbf{cause}$). Although **corresp′** only contains one term like this, there is a **jane** component in **corresp** for every pattern which is shared by **P** and $\mathbf{E_{CM}}$, which adds up to a very strong component of **jane** in **corresp**. When **corresp** is compared to the fillers of $\mathbf{E_{CM}}$, **corresp** is more similar to **fred** than **rover**, due to the strong **jane** component in **corresp**.

One way of eliminating this similar-class bias is to perform a linear, multi-way, role-clean-up on **jane-roles**. This should pass all positive role components and suppress negative role and non-role components like $\mathbf{cause}\ \ \mathbf{jane}^T$. Thus, the clean version of **jane-roles** is as follows:

$\mathbf{clean\text{-}jane\text{-}roles} = 0.20\ \mathbf{cause}_{antc} + 0.18\ \mathbf{cause}_{cnsq} + 0.13\ \ \mathbf{flee}_{agt} + 0.12\ \mathbf{bite}_{obj}$

This clean-up can be viewed as a less competitive version of the standard clean-up. The corresponding objects extracted using role clean-up are shown in Table 4. This process gives the correct answers for the three episodes where there is a consistent mapping.

$\mathbf{E_{LS}}$ **cleaned-jane-roles**^{T}	**john**	0.27	✓
	fido	0.20	

$\mathbf{E_{CM}}$ **cleaned-jane-roles**^{T}	**fred**	0.20	
	rover	0.29	✓
$\mathbf{E_{AN}}$ **cleaned-jane-roles**^{T}	**felix**	0.25	✓
	mort	0.20	
$\mathbf{E_{SF}}$ **cleaned-jane-roles**^{T}	**john**	0.25	?
	fido	0.17	
$\mathbf{E_{FOR}}$ **cleaned-jane-roles**^{T}	**mort**	0.26	?
	felix	0.19	

Table 4.

The other way of avoiding the similar-class bias is to use a different binding operation, in which the algebraic properties of encoding and decoding do not result in terms like (**cause jane**^{T})^{T} **cause** equating to **jane**. Two possible suitable alternative binding operations are convolution of permuted vectors, and random sums of pairwise products (Plate, 1994).

There are two more limitations with these fast techniques for deriving interpretations. One is that each corresponding pair in a mapping is extracted independently. This matters when there is more than one consistent mapping. For example, if we have two possible consistent mappings $\{X \leftrightarrow A,\ Y \leftrightarrow B\}$ and $\{X \leftrightarrow B,\ Y \leftrightarrow A\}$, then the choice of mapping for X should constrain the choice for Y, but this will not be the case with the above techniques. To overcome this problem requires some other mechanism for checking that a mapping is one-to-one. The other problem is that these techniques fail when two different objects have the same set of roles – in such a case ambiguous results can be produced.

6 Chunking & Memory Organization

HRRs provide a natural method for chunking, i.e., breaking large structures into pieces of manageable size and indexing them. In fact, to store structures of unlimited size, a model based on HRRs must store sub-structures (i.e., chunks) in clean-up memory (see Section 3.7).

As more items and bindings are stored in a single HRR the quality of the items that can be extracted drops. If too many component patterns are stored the quality will be so low that the extracted items will be easily confused with similar items or, in worse cases, completely unrecognizable. The number of items and bindings in a HRR grows with both the height and width of the structure being represented. For example, the skeleton HRRs $\mathbf{K_{P\text{-}bite}}$ and $\mathbf{K_P}$ have 3 and 7 component patterns respectively, while the full HRRs $\mathbf{P_{bite}}$ and **P** (the structural skeletons plus filler patterns) have 5 and 19 component patterns. Since the number of items and bindings that can be stored for a given quality of decoding grows linearly with the vector dimension (Plate 1995), storing large structures which might have hundreds or more components would require extremely high dimensional vectors. A superior approach to storing large structures is to break the structure into chunks. This requires storing sub-structures in the item memory, and using them when decoding components of complex structures. For example, to decode the agent of the cause antecedent of **P** we first extract the cause antecedent pattern. This gives a noisy version of $\mathbf{P_{bite}}$, which can be cleaned up by accessing item memory and retrieving the closest match. Now we have an accurate version of $\mathbf{P_{bite}}$ from which we can extract the filler of the agent role. Using chunks during decoding means that only top-level items and bindings count towards the total effective number of items and bindings. This makes it possible to store structures of unlimited size.

To use chunks there must be a way of referring, or pointing to the chunks. In content-addressable memory in general, "pointers" to sub-chunks cannot be addresses, but must somehow hint at the contents of the sub-chunk. In HRRs, a decoded filler or sub-chunk, which is derived from a chunk by decoding with a role pattern, functions as an associative "pointer" to a pattern in item memory. These associative pointers are different from conventional pointers in that their form conveys information about their referent, information that is noisy but immediately available without the need to access memory. The advantage of having pointers that encode information about their referents is that some operations can be performed without following the pointer. This can save much time. For example, we can decode nested fillers quickly if very noisy results are acceptable, or we can get an estimate of the similarity of two structures without decoding them.

6.1 Overall memory organization

In a system that uses HRRs there must be two levels of memory organization. One level encodes the structure in and among chunks. The other level stores large numbers of chunks (the large-scale clean-up memory).

<Figure 2 about here>

Convolution encoding is most suited for encoding structure in and among small chunks in memory. Figure 2 shows the contents of large-scale memory when several chunked items are stored, and the relationships among the chunks. Each rectangle is a pattern stored in long term memory. Arrows show sub-chunk relationships and are labelled with the role-pattern (in italics) through which the sub-chunk can be found. The two top level chunks shown here are **P** and $\mathbf{E}_{\mathbf{LS}}$ from earlier. These contain lower level chunks like $\mathbf{P}_{\mathbf{bite}}$ (Spot bit Jane) and $\mathbf{P}_{\mathbf{flee}}$ (Jane fled from Spot). Decoding fillers via role vectors is one way of navigating among related chunks. Similarity can also be used to traverse memory: $\mathbf{P}_{\mathbf{bite}}$ is similar to **bite**, **spot** and **jane**, though these "links" are not shown in the figure.

Because of its memory capacity characteristics and noise in retrieval, convolution does not provide a suitable associative memory technique for the clean-up memory, which must store all the chunks. For this purpose we require some sort of large-scale error-correcting auto-associative memory. This large-scale memory should have the following properties:

auto-associative & error-correcting ability: when given a pattern, it should return accurately the closest one(s) stored, and

high capacity: the number of patterns which can be stored should be exponential in the size of the patterns.

There are several ways the clean-up memory could be implemented, e.g., Kanerva's (1988) sparse distributed memory, and Baum, Moody and Wilczek's (1988) various associative content addressable memory schemes. The simplest of the latter schemes, which is perfectly adequate for the purposes of HRRs, is the unary "grandmother cell" memory. This type of memory can be thought of as a feedforward neural network with one hidden layer. A hidden unit (the grandmother cell) is devoted to each pattern stored. Incoming and outgoing weights on a hidden unit are the pattern that unit stores, and the activity of hidden units is the result of a winner-take-all competition. Thus, when the network is presented with a noisy pattern, the hidden unit which stores the closest pattern becomes active and propagates the clean version of the pattern to the output. This type of network can be straightforwardly implemented in parallel hardware and also can be easily and efficiently simulated on a serial computer by storing vectors in a list and doing an exhaustive search for the closest match.

7 Discussion

This paper has described a scheme for encoding structure in vector representations based on circular convolution. Other approaches, such as Smolensky's (1990) tensor products, Pollack's (1990) RAAMs, Kanerva's (1996) binary spattercodes, and Gayler's (1998) braided codes have much in common – see Plate (1997) for a discussion – and could also be used in models of analogy processing.

The origin of patterns representing classes such as 'dog', 'cat' and 'human' must be addressed at some stage. One possible automatic technique for learning such patterns is Latent Semantic Analysis (LSA), which learns high-dimensional vector patterns for words from large quantities of text (Landauer, Laham, and Foltz 1998). These patterns reflect human similarity judgements and could easily be used with HRRs.

The existence of a fast technique for computing good guesses at object correspondences suggests a new model for analogical mapping. Mapping could be done by "guessing" correspondences while stepping through the components of two structures and verifying that the proposed correspondences are consistent. This would require three mechanisms, one for traversing structures, another for guessing correspondences, and the last for storing correspondences and checking their consistency. All can be implemented with operations on vector representations. Such a model differs from ACME and SME in that it puts complexity at a different level. The top level involves simple sequential computation (traversing a structure and checking for mapping inconsistencies) rather than complex structural matching or con-

struction of special networks, while the bottom level involves information-rich vector processing to measure similarities and estimate correspondences.

8 Conclusion

Holographic Reduced Representations provide a useful vector representation for analog retrieval and processing tasks. They provide chunking, which will be essential in vector-based model that stores large structures. They also afford fast operations for computing similarity and object correspondences. These fast operations appear to have the right amount of power for modeling human abilities: their strengths and weaknesses follow a similar pattern to human performance on similarity estimation and mapping. In particular, HRRs provide a simple, single-stage model of human performance on analog retrieval: HRR dot-products are sensitive to superficial similarity, and also to structural similarity in situations where corresponding roles have similar fillers, which is the same pattern of performance as observed in human subjects on analog retrieval tasks.

9 Acknowledgements

Thanks to the two anonymous referees who provided very helpful comments.

10 References

ANDERSON J R 1983 *The architecture of cognition*. Cambridge, Mass.: Harvard University Press.

BAUM E B, MOODY J and WILCZEK F 1988 Internal representations for associative memory. *Biological Cybernetics* 59, 217–228.

BORSELLINO A and POGGIO T 1973 Convolution and correlation algebras. *Kybernetik* 13, 113–122.

FODOR J A and PYLYSHYN Z W 1988 Connectionism and cognitive architecture: A critical analysis. *Cognition* 28, 3–71.

FORBUS K D, GENTNER D and LAW K 1994 MAC/FAC: A model of similarity-based retrieval. *Cognitive Science* 19, 141–205.

GAYLER R W 1998 Multiplicative Binding, Representation Operators and Analogy. In HOLYOAK K and GENTNER D and KOKINOV B (Eds.), *Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences.* New Bulgarian University, Sofia, Bulgaria, p405. Note: only the abstract appears in the proceedings; full text is available at http://cogprints.soton.ac.uk

GENTNER D and MARKMAN A B 1993 Analogy – Watershed or Waterloo? Structural alignment and the development of connectionist models of analogy. In GILES C L, HANSON S J, and COWAN J D (Eds.), *Advances in Neural Information Processing Systems 5* (NIPS*92), San Mateo, CA, 855–862. Morgan Kaufmann.

GENTNER D, RATTERMANN M J, and FORBUS K D 1993 The roles of similarity in transfer: Separating retrievability from inferential soundness. *Cognitive Psychology* 25(4), 524–575.

GENTNER D and TOUPIN C 1986 Systematicity and surface similarity in the development of analogy. *Cognitive Science* 10(3), 277–300.

HINTON G E 1990 Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence* 46(1-2), 47–76.

HINTON G E, MCCLELLAND J L, and RUMELHART D E 1986 Distributed representations. In RUMELHART D E, MCCLELLAND J L, and the PDP research group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*, Volume 1, 77–109. Cambridge, MA: MIT Press.

HOLYOAK K J and KOH K 1987 Surface and structural similarity in analogical transfer. *Memory and Cognition* 15, 332–340.

HUMMEL J E and HOLYOAK K J 1997 Distributed representations of structure: A theory of analogical access and mapping. Psychological Review 104(3), 427–466.

KANERVA P 1988 *Sparse Distributed Memory*. Cambridge, MA: MIT Press.

KANERVA P 1996 Binary spatter-coding of ordered k-tuples. In VON DER MALSBURG C, VON SEELEN W, VORBRUGGEN J, and SENDHOFF B (Eds.), Artificial Neural Networks–ICANN Proceedings, Volume 1112 of *Lecture Notes in Computer Science, Berlin*, 869–873. Springer.

KEANE M 1988). *Analogical Problem Solving*. New York: Wiley.

LANDAUER T K, LAHAM D, and FOLTZ P W 1998 Learning Human-like Knowledge with Singular Value Decomposition: A Progress Report. In *Neural Information Processing Systems* (NIPS*97). MIT Press.

LIEPA P 1977 Models of content addressable distributed associative memory. Unpublished manuscript.

MCCLELLAND J L and RUMELHART D E 1981 An interactive activation model of context effects in letter perception, part I: An account of basic findings. *Psychological Review* 88, 375–407.

METCALFE EICH J 1982 A composite holographic associative recall model. *Psychological Review* 89, 627–661.

MURDOCK B B 1982 A theory for the storage and retrieval of item and associative information. *Psychological Review* 89(6), 316–338.

MURDOCK B B 1987 Serial-order effects in a distributed-memory model. In GORFEIN D S and HOFFMAN R R (Eds.), *MEMORY AND LEARNING: The Ebbinghaus Centennial Conference*, 277–310. Lawrence Erlbaum Associates.

PAEK E G and PSALTIS D 1987 Optical associative memory using Fourier transform holograms. *Optical Engineering* 26(5), 428–433.

PLATE T A 1994 Distributed Representations and Nested Compositional Structure. Ph.D. thesis, Department of Computer Science, University of Toronto. Available at http://cogprints.soton.ac.uk.

PLATE T A 1995 Holographic reduced representations. *IEEE Transactions on Neural Networks* 6(3), 623–641.

POLLACK J B 1990 Recursive distributed representations. *Artificial Intelli*gence 46(1-2), 77–105.

RATCLIFF R and MCKOON G 1989 Similarity information versus relational information: Differences in the time course of retrieval. *Cognitive Psychology* 21, 138–155.

ROSS B 1987 This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 13(4), 629–639.

ROSS B 1989 Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15(2), 456–468.

SCHÖNEMANN P H 1987 Some algebraic relations between involutions, convolutions, and correlations, with applications to holographic memories. *Biological Cybernetics* 56, 367–374.

SEIFERT C M, MCKOON G, ABELSON R P, and RATCLIFF R 1986 Memory connections between thematically similar episodes. *Journal of Experimental Psychology*: Learning, Memory, and Cognition 12, 220–231.

SMOLENSKY P 1986 Neural and conceptual interpretation of PDP models. In MCCLELLAND J L, RUMELHART D E, and the PDP research group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition*, Volume 2, 390–431. Cambridge, MA: MIT Press.

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

THAGARD P, HOLYOAK K J, NELSON G, and GOCHFELD D 1990 Analog Retrieval by Constraint Satisfaction. *Artificial Intelligence* 46, 259–310.

WHARTON C M, HOLYOAK K J, DOWNING P E, LANGE T E, WICKENS T D, and MELZ E R 1994 Below the surface: Analogical similarity and retrieval competition in reminding. *Cognitive Psychology* 26, 64–101.

WILLSHAW, D. (1981). Holography, associative memory, and inductive generalization. In G. E. Hinton and J. A. Anderson (Eds.), *Parallel models of associative memory*. Hillsdale, NJ: Erlbaum.

WILLSHAW D, BUNEMAN O P, and LONGUET-HIGGINS H C 1969 Non-holographic associative memory. *Nature* 222, 960–962