Representation and Processing of Structures with Binary Sparse Distributed Codes

Dmitri A. Rachkovskij

Abstract—The schemes for compositional distributed representations include those allowing on-the-fly construction of fixed dimensionality codevectors to encode structures of various complexity. Similarity of such codevectors takes into account both structural and semantic similarity of represented structures. In this paper, we provide a comparative description of sparse binary distributed representation developed in the framework of the associative-projective neural network architecture and the more well-known holographic reduced representations of Plate and binary spatter codes of Kanerva. The key procedure in associative-projective neural networks is context-dependent thinning which binds codevectors and maintains their sparseness. The codevectors are stored in structured memory array which can be realized as distributed auto-associative memory. Examples of distributed representation of structured data are given. Fast estimation of the similarity of analogical episodes by the overlap of their codevectors is used in the modeling of analogical reasoning both for retrieval of analogs from memory and for analogical mapping.

Index Terms—Sparse coding, binary coding, binding, representation of structure, hierarchical representation, nested representation, long-term memory, analogy, compositional distributed representations, connectionist symbol processing, analogical retrieval, analogical mapping.

1 Introduction

The problem of representation is of great importance in Artificial Intelligence (AI). In localist connectionist models, each item is represented by a single unit, node, or neuron (at least in long-term memory). Such representations are criticized in respect to their limited information capacity and insufficient semantic sensitivity.

For example, in order to represent nested compositional structures, it is necessary to represent their constituents, various compositions of those constituents, combinations of those compositions, etc. For a significant number of complex recursive structures, the number of possible combinations and localist elements required to represent them becomes prohibitive because of exponential growth.

At the same time, a local representational element (or symbol) does not carry immediate information about its constituents and acts just as a meaningless pointer. For example, in order to estimate the similarity of hierarchical structures encoded with localist representations, it is generally necessary to find and match their constituents down to the lowest hierarchical level using pointer (or connection) following.

In fully distributed representations, each item is encoded by a pattern of activity over a pool of units. The number of different nonorthogonal patterns exponentially exceeds the dimensionality of the coding pool (see, e.g., [42]). It allows for a high information capacity of distributed representations. An estimation of item similarity can be realized by the dot product of the vectors corresponding to their activity patterns.

Distributed representations are sometimes perceived as unsuitable for hierarchical (recursive, nested) structure representation. However, in fact, a number of schemes have been proposed for distributed representation of structured data, including BoltzCONS of Touretzky [55], reduced descriptions of Hinton [20], tensor products of Smolensky [51], RAAMs of Pollack [43], LRAAMs of Sperduti [52] (see also [41] for a review). Of special interest to us are the schemes where the codes representing new structures of various complexity are constructed on-the-fly (without special learning) and have the same dimensionality. Associative-Projective Neural Networks (APNNs) [29], Holographic Reduced Representations (HRRs) [40], and Binary Spatter Codes (BSCs) [26] are among such schemes.

The ability to estimate easily the similarity of complex structured representations is essential for the solution of many AI problems [29]. One of these problems is the modeling of human analogy-making. This requires taking into account the structure and semantics of analogs. Due to the common supposition that distributed representations poorly deal with structure, it is natural that most influential computational models of analogical reasoning (ARCS-ACME [53], [22], MAC/FAC-SME [9], [7], Copycat [21]) are essentially based on symbolic or localist representations.

However, the quest to enhance the semantic basis of analog representations and to reach a higher degree of neurological relevancy leads to the attempts for augmenting the models of analogy with some share of distributed representations (LISA [24], STAR2 [16], DRAMA [6]). Operations on exclusively distributed representations useful for analogical processing were demonstrated by Plate using HRRs [39], [42] and Kanerva using BSCs [27], [28].

• *The author is with the V.M. Glushkov Cybernetics Center, Prospect Glushkova 40, Kiev 03680, Ukraine. E-mail: dar@infrm.kiev.ua.*

Manuscript received 19 July 1999; revised 10 Jan. 2000; accepted 14 Feb. 2000.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 112685.

In this paper, an approach to the representation and manipulation of structures using binary sparse distributed codes and techniques of APNNs is considered and compared with the approaches adopted in HRRs and BSCs. In Section 2, we provide a comparative description of basic features and operations of APNNs, HRRs, and BSCs. Various schemes for the distributed representation of structures are presented in Section 3. Some considerations concerning the organization and implementation of long-term memory for structures are given in Section 4. In Section 5, we apply APNN-style representations for the retrieval and mapping of toy analogical episodes and compare the results with those of Plate. Sections 6 and 7 present a discussion and conclusion correspondingly.

2 On-the-Fly Distributed Representations of Structure: Basic Features and Operations

In order to represent nested compositional structures efficiently, the following features of APNN-style representations are essential:

- Representations should be fully distributed. An item of any complexity or nesting level, including the lowest (base) level items, is represented by a distributed activity pattern over the pool of units. Thus, the number of representable items is not limited by the number of available units.
- Dimensionality should be preserved. The dimensionality of the coding units' pool for items of any complexity or nesting level is the same. This avoids the problems concerning the growth of the composite items' code dimensionality.
- Representations should be constructed "on-the-fly." The code of a composite item of any complexity or nesting level is formed from the codes of its constituents without training. This allows a fast recursive construction of representations of arbitrary items.
- Similar items should produce similar representations. This allows the similarity of items to be estimated by the dot product of their codevectors. The degree of similarity varies gradually.

These requirements are met, apart from APNNs, by HRRs and by BSCs. Therefore, in this section, we consider the similarities and the differences between these schemes. For comparison of APNNs with other distributed schemes for structure representation, see [46].

2.1 The Nature of Distributed Representations

In fully distributed representations, each item (a feature, an object, a relation, a structure, etc.) is encoded by an activity pattern over the total pool of units. It is convenient to represent the pool of N units by an N-dimensional vector and the activity of each unit by the value of the corresponding vector element. Let us denote codevectors by a boldface font and the items by a cursive font.

In HRRs, elements of codevectors are real numbers with the Gaussian distribution of mean 0 and variance $1/N$. For example, $N = 2,048$ [42]. Similarity of the HRR codevectors is calculated as their dot product.

The BSCs are binary codevectors (or "codewords" in terms of Kanerva [26]). Each vector element can be 0 or 1 with the probability 0.5. Let us denote the number of 1s in a codevector by M. The density of a codevector $\mathbf{X}$ can be defined as the probability of 1s in it: $p(\mathbf{X}) = M/N$. Therefore, the codevectors of BSCs are "dense:" $p(\mathbf{X}) = 0.5$. For example, $N = 10,000$, $M = 5,000$ [27]. The distance between two codevectors is defined as the Hamming distance. Calculation of the codevectors' similarity by the dot product actually counts the number of coinciding 1s.

Code vectors in the APNN scheme are sparse binary vectors: $p(\mathbf{X}) \ll 1.0$. It is closer to neurologic data and useful for associative memory capacity. For example, $p(\mathbf{X}) = 0.01$, $N = 100,000$, and $M = 1,000$. The similarity of two APNN codevectors is estimated by the overlap of their 1s, usually normalized to the number of 1s in one of the codevectors.

Usually, the elements of codevectors corresponding to dissimilar items of the lowest (base) hierarchical level are generated randomly and independently. Generated codevectors are stored. Encoding of the composite structures will be considered below.

2.2 Superposition

A set of items is represented by the superposition of their codevectors. In HRRs, superposition is realized by an elementwise addition. In BSCs, it is realized by bitwise addition of codevectors (with subsequent thresholding for binarization). In APNNs, the codevector's superposition is implemented by bitwise disjunction. Obviously, for all these types of superposition, the result preserves the dimensionality of initial codevectors.

Superposition also preserves the similarity of the result to each of the superimposed codevectors. For independent codevectors, $\mathbf{D} = \mathbf{A} + \mathbf{B}$ is similar to $\mathbf{A}$ and $\mathbf{B}$ ("+" denotes superposition). This type of similarity is called "unstructured" similarity [41]. The similarity of codevectors to their superposition decreases as the number of superimposed codevectors increases. Therefore, superposition similarity can be also considered as "additive."

Unstructured similarity allows one to determine which codevectors are superimposed by finding the codevectors (from the set of component codevectors) that are most similar to the superposition codevectors.

2.3 Binding

2.3.1 Necessity of Binding and Some Related Problems

Superposition alone is not enough to encode structures. Under superposition, the components of structures are mixed together and the information on their combination or order in substructures is not preserved. Composition by superposition is the reason that distributed representations are claimed to lack structure. Two common formulations of this problem are "ghosts" and "superposition catastrophe" ([8], [57], see also [46]).

The simplest illustration could be as follows: Let there be component items A, B, and C and composite items AB, AC, and BC, which are encoded by the simultaneous "all-or-none" activation of component item representations (Fig. 1). If representations of any two of three composite items (e.g., AB and BC) are activated, the third

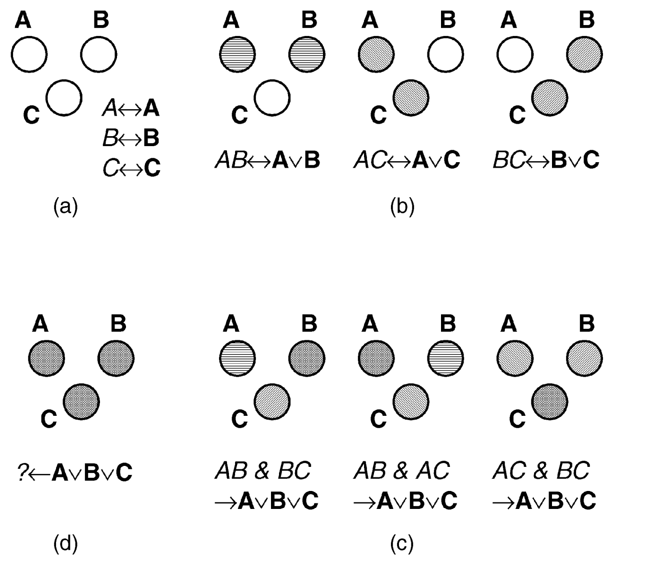

Fig. 1. An illustration of "ghosts" or "superposition catastrophe." (a) Each component item A, B, and C is represented by the "all-or-none" activity pattern $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, respectively. (b) Each composite item AB, AC, and BC is represented by the superposition of activity patterns of the component items. (c) Superposition of activity patterns of any two of three composite items produces the representation in which the third unforeseen composite item (ghost) is encoded as well. (d) If all three component items are activated, it is impossible to tell which composite items are actually present (superposition catastrophe).

composite item will become active as well, though unbidden (here, "ghost" AC). In "superposition catastrophe" formulation, the same situation is described as follows: Which two composite items are really present if representations of component items A, B, and C are activated? A related problem is that it is impossible to encode the order or sequence of items by their simultaneous activation: The pattern for ABC is indistinguishable from the patterns for CBA or BCA.

It should be noted that these problems persist both for localist and distributed representations of the component items A, B, and C, if the presence of a composite item is encoded by the simultaneous activation of its component representations. These problems also manifest themselves as "spurious memories" ([23], [56]) for superposition learning rules, e.g., Hebb's learning rule [19] for matrix-type distributed associative memory (i.e., spurious attractors in Willshaw or Hopfield networks).

Thus, in order to represent compositional structures, one should be able to represent "bindings" of component items. In localist models, such bindings are represented by the introduction or allocation of new units (e.g., the AB unit to represent the binding of A and B). This increases the dimensionality of the unit pool when new bindings and structures are introduced (Fig. 2).

A local node itself does not carry any information concerning the composite item it represents. Therefore, between-unit connections are needed to point to the nodes representing the constituents. Thus, in order to estimate the similarity (just in terms of common base-level components) of two structures represented by local nodes, both of them must be unfolded through the nodes representing their base-level components (Fig. 3).

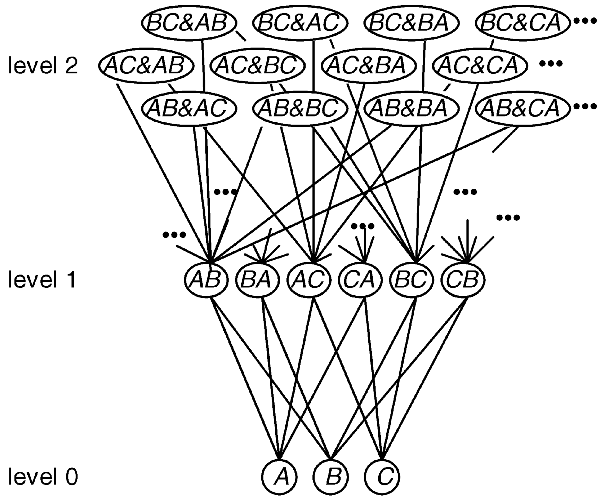

Fig. 2. Growth of the number of units which represent possible compositional structures versus the level of compositional hierarchy.

If the comparison should take into account the structural arrangement of components, much more complicated techniques are needed, as exemplified by the models of analogy processing, where analogs are structured propositions (Fig. 4). These techniques include the construction and settling of special constraint satisfaction networks which find the best match between the corresponding items of analogs (ACME [22], DRAMA [6]), symbolic processing of virtual networks representing those correspondences (SME [7]), or use of "mapping connections" between "dynamic bindings" ([36], [48]) represented by the temporal coactivation of units in existing localist network (LISA [24]).

In the distributed representations of structure we consider here, the binding of codevectors is implemented by special operations which do not change dimensionality. This is important for the representation of nested structures,

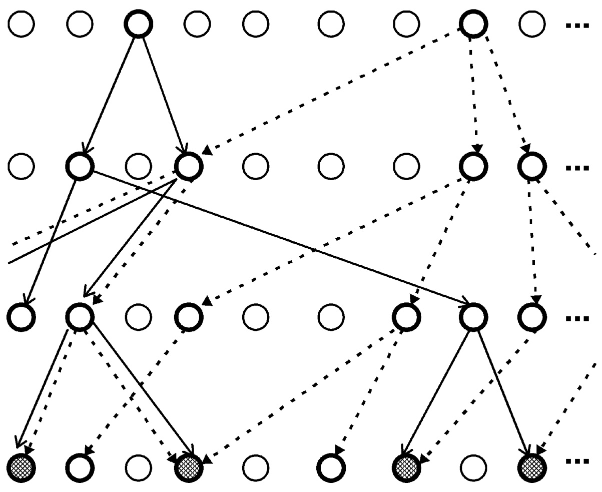

Fig. 3. In order to find the similarity of the base-level components of two locally represented hierarchical structures (even without the structural similarity), the nodes representing the whole structure at the top hierarchical level should be unfolded through the component nodes of the lower levels (down to the base-level) using connections between nodes. The solid and dotted arrows show the unfolding of two composite items. The shaded nodes at the lowest level correspond to common base-level components.

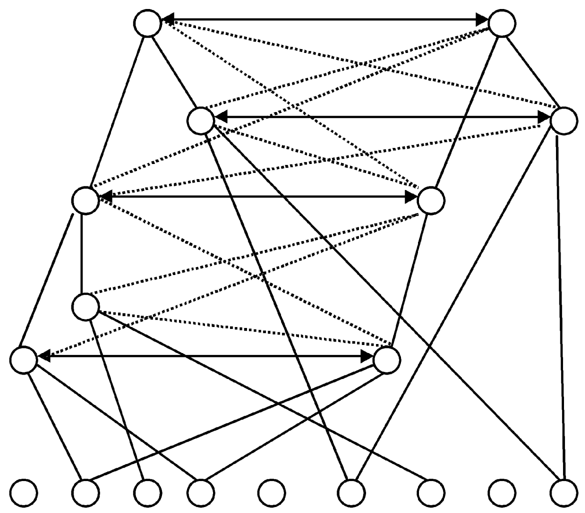
Fig. 4. An estimation of the total similarity (including structural similarity) of locally represented hierarchical structures requires computationally expensive alignment of their components. The arrows show probably the best match between the component nodes, and the dotted lines show some of the other possible candidate correspondences. The plain lines show "part-of" connections between the lower- and the higher-level nodes.

where the bindings of lower hierarchical levels are used as the components of higher-level bindings.

2.3.2 Binding in HRRs and BSCs

Circular convolution is used in HRRs to obtain the codevector **Z** that represents the binding of two codevectors **X** and **Y**:

$$\mathrm{Z_i} = \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{X_j Y_{i-j}}, \tag{1}$$

where subscripts are taken modulo-N. This operation can be also written as $\mathbf{Z} = \mathbf{X} * \mathbf{Y}$, where "$*$" denotes circular convolution. Binding by circular convolution preserves the Gaussian distribution of codevector elements.

Unlike the result of superposition, the result **Z** of binding obtained by circular convolution is generally dissimilar to the codevectors **X** and **Y**. However, another kind of similarity is preserved. $\mathbf{D} = \mathbf{A} * \mathbf{B}$ is similar to $\mathbf{D}' = \mathbf{A}' * \mathbf{B}'$, if **A** is similar to $\mathbf{A}'$ and **B** is similar to $\mathbf{B}'$. This kind of similarity is called "structured" similarity [41]. Binding by circular convolution preserves structured similarity in a multiplicative fashion. If **A** and **B** are independent, as well as $\mathbf{A}'$ and $\mathbf{B}'$, then the similarity of **D** to $\mathbf{D}'$ is equal to the product of the similarities of **A** to $\mathbf{A}'$ and **B** to $\mathbf{B}'$, e.g., if $\mathbf{A} = \mathbf{A}'$, the similarity of **D** to $\mathbf{D}'$ will be equal to the similarity of **B** to $\mathbf{B}'$. If **A** is not similar to $\mathbf{A}'$, **D** will have no similarity with $\mathbf{D}'$ irrespective of the similarity between **B** and $\mathbf{B}'$.

This type of similarity is different from the structured similarity of superimposed codevectors: $\mathbf{D} = \mathbf{A} + \mathbf{B}$ will still be similar to $\mathbf{E} = \mathbf{A} + \mathbf{C}$ if **B** is dissimilar to **C**.

Gayler [12] proposed to use elementwise multiplication to bind real-valued codevectors. Binding in BSCs is implemented by elementwise XOR of codevectors. These binding operations preserve the similarity in approximately the same manner as circular convolution.

2.3.3 Binding in APNNs

In APNNs, binding is realized by the Context-Dependent Thinning (CDT) procedure. The first version of this procedure was proposed by Kussul under the name "normalization" (e.g., [30], [33]). Various versions of the CDT (thinning) procedure and their algorithmic and neural network implementations are discussed in [46].

Here, we will describe the "additive" version of the CDT procedure [46]. In this version, a single codevector **Z** is input to the procedure. This codevector is the disjunctive superposition (Section 2.2) of two or several component codevectors to be bound:

$$\mathbf{Z} = \bigvee_{s=1}^{S} \mathbf{X_s}, \tag{2}$$

where S is the number of components and $\mathbf{X_s}$ is the (randomly generated) codevector of the sth component. Then,

$$\langle \mathbf{Z} \rangle = \bigvee_{k=1}^{K} (\mathbf{Z} \wedge \mathbf{Z}^{\sim}(k)) = \mathbf{Z} \wedge \bigvee_{k=1}^{K} \mathbf{Z}^{\sim}(k). \tag{3}$$

Here, $\mathbf{Z}^{\sim}(k)$ is **Z** with permuted elements, $\langle \mathbf{Z} \rangle$ is the thinned codevector. (In the framework of HRRs, Plate uses angle brackets $\langle ... \rangle$ to denote "normalization," see Section 2.5). Each kth permutation must be fixed, unique, and independent. Random permutations would be ideal, however, cyclic shift permutations with a random number of shifts are convenient to use in implementations. Also, some permutations may even coincide accidentally—they just would not add 1s into disjunction of (3).

The CDT procedure reduces the density of the input codevector **Z** down to some specified value. Usually, it is comparable to the density of its component codevectors $\mathbf{X_s}$. The number K of permuted and disjunctively superimposed vectors is chosen so that the number of 1s in $\langle \mathbf{Z} \rangle$ becomes approximately the number that is needed. For example, let the density of each component codevector be

$$p(\mathbf{X_s}) = 0.01 \ \ \forall s$$

and let us bind three components ($S = 3$). Since the component codevectors are sparse and independent, they have a statistically small overlap:

$$p(\mathbf{X_s} \wedge \mathbf{X_r}) = p(\mathbf{X_s}) * p(\mathbf{X_r}) = 0.01 * 0.01 = 0.0001 \ (r \neq s). \tag{4}$$

Therefore, the density of the input superposition codevector is approximately

$$p(\mathbf{Z}) \approx p(\mathbf{X_s}) * S = 0.01 * 3 = 0.03. \tag{5}$$

$\mathbf{Z}^{\sim}$ and **Z** are independent because of permutation. Therefore, the probability of 1s in their conjunction is

$$p(\mathbf{Z} \wedge \mathbf{Z}^{\sim}) = p(\mathbf{Z}) p(\mathbf{Z}^{\sim}) = 0.03 * 0.03 = 0.0009. \tag{6}$$

Thus, we need to superimpose by disjunction approximately $K \approx p(\mathbf{X_s})/p(\mathbf{Z} \wedge \mathbf{Z}^{\sim}(k)) \approx 11$ of $\mathbf{Z} \wedge \mathbf{Z}^{\sim}(k)$ in order to get

$$p(\langle \mathbf{Z} \rangle) \approx p(\mathbf{X_s}).$$

bit#	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{Z} \wedge (\mathbf{Z}^{\sim}(1) \vee \mathbf{Z}^{\sim}(2) \vee \mathbf{Z}^{\sim}(3)) = \langle \mathbf{Z} \rangle$				
1	1	0	1	0	0	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	1	0
4	0	1	1	0	0	0	0
5	0	0	0	1	1	0	0
6	0	1	1	0	0	1	1
7	0	0	0	0	1	0	0
8	1	0	1	1	0	1	1
9	0	0	0	0	1	0	0
10	0	0	0	1	0	1	0
11	0	0	0	0	0	0	0
12	0	0	0	1	0	0	0

Fig. 5. Toy example of the additive CDT procedure. Twelve-bit codevectors $\mathbf{X_1}$ and $\mathbf{X_2}$ are first superimposed in $\mathbf{Z}$. Then, $\mathbf{Z}$ is conjuncted with its shift permutations (4-bit shift, 1-bit shift, 2-bit shift down) until the number of 1s accumulated in the resulting thinned codevector $\langle \mathbf{Z} \rangle$ reaches a predefined value. In our case, the number of 1s in $\langle \mathbf{Z} \rangle$ is equal to the number of 1s in each of the components $\mathbf{X}_1$ and $\mathbf{X}_2$.

Let us consider a toy example of thinning (Fig. 5). Two component codevectors $\mathbf{X_1}$ and $\mathbf{X_2}$ each have $m = 2$ 1s in a total of $n = 12$ bits. (Remember, that actually $M \approx 1{,}000$ and $N = 100{,}000$). $\mathbf{Z} = \mathbf{X_1} \vee \mathbf{X_2}$ has four 1s. Its first permutation, $\mathbf{Z}^{\sim}(1)$, is obtained as a 4-bit down shift. It can be seen that $\mathbf{Z} \wedge \mathbf{Z}^{\sim}(1)$ produces a single 1. $\mathbf{Z} \wedge \mathbf{Z}^{\sim}(2)$, where $\mathbf{Z}^{\sim}(2)$ is a 1-bit down shift of $\mathbf{Z}$, produces no 1s. However, after $\mathbf{Z}^{\sim}(3)$ is obtained as a 2-bit down shift of $\mathbf{Z}$, $m = 2$ 1s are accumulated in $\langle \mathbf{Z} \rangle$ and we stop the thinning procedure.

The algorithm of the additive CDT procedure [46] is shown in Fig. 6. A lot of orthogonal "patterns" of thinning (disjunctions of different permutations, such as $\mathbf{Z}^{\sim}(1) \vee \mathbf{Z}^{\sim}(2) \vee \mathbf{Z}^{\sim}(3)$ in Fig. 5) are possible depending on the seed in the algorithmic implementation. We will denote different thinning patterns by different labels shown as the superscript of the left angle bracket, e.g., ${}^{1}\langle \mathbf{Z} \rangle$, ${}^{2}\langle \mathbf{Z} \rangle$. ${}^{5}\langle \mathbf{Z} \rangle$, ${}^{u}\langle \mathbf{Z} \rangle$, where 1, 2, 5, u are the labels denoting different thinning patterns. Inside each level of compositional hierarchy (see Section 4.1), the thinning pattern is permanent.

```
The input codevector $\mathbf{Z} = \mathbf{X}_1 \vee \mathbf{X}_2 \vee \ldots \vee \mathbf{X}_S$
Set the output (thinned) codevector $\mathbf{T}$ to 0.
Seed the random-number generator:
randomize(*seed*).

while(|$\mathbf{T}$| < M)
        if ((r=rand()) $\neq$ 0)
                for(i=1; 2, …, N)
                        $T_i = T_i \vee (Z_i \wedge Z_{i+r \text{ modulo } N})$
```

Fig. 6. An example of an algorithmic implementation of the additive version of the Context-Dependent Thinning procedure. Parameter *seed* defines the configuration of shift permutations. This configuration (and *seed*) is usually constant for a given level of part-whole hierarchy. Different (but constant) seeds are usually used for different levels of part-whole hierarchy.

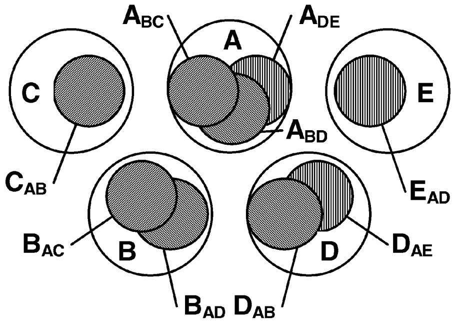

Fig. 7. How thinning preserves similarity. Big clear circles represent the 1s encoding the component items A, B, C, D, and E. Representations of three composite items ABD, ABC, and ADE formed by thinning are shown by the corresponding combinations of three small differently shaded circles. $\mathbf{X_{YZ}}$ denotes the subset of 1s preserved in the thinned representation of item $\mathbf{X}$ when items Y and Z are also present. It can be seen that $\mathbf{A_{BC}}$, $\mathbf{A_{BD}}$, and $\mathbf{A_{DE}}$ are all different subsets of $\mathbf{A}$, and $\mathbf{A_{BC}}$ is more similar to $\mathbf{A_{BD}}$ than to $\mathbf{A_{DE}}$.

Unlike the binding operations in HRRs and BSCs, thinning in APNNs preserves both unstructured and structured similarity of the codes. Thinning preserves a subset of the 1s of $\mathbf{Z}$ (3). Therefore, the thinned codevector is similar to the codevector of all components superimposed in the input codevector, thus preserving unstructured similarity.

The fraction of 1s in the output thinned codevector from each component codevector is proportional to the density of the component codevectors. For example, for independent $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of the same density,

$$\langle \mathbf{Z} \rangle = \langle \mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \rangle; p(\langle \mathbf{Z} \rangle \wedge \mathbf{A}) = p(\langle \mathbf{Z} \rangle \wedge \mathbf{B}) \\ = p(\langle \mathbf{Z} \rangle \wedge \mathbf{C}) = p(\langle \mathbf{Z} \rangle)/3.$$

If $p(\langle \mathbf{Z} \rangle) = p(\mathbf{A}) = p(\mathbf{B}) = p(\mathbf{C})$, approximately 0.33 of 1s will remain from each of the component codevectors.

However, the subset of 1s preserved in the thinned codevector from each component depends on the other components in the input superposition. For the example of three-component items, the subset of 1s from $\mathbf{A}$ preserved in $\langle \mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \rangle$ is different from the subset of 1s from $\mathbf{A}$ preserved in $\langle \mathbf{A} \vee \mathbf{B} \vee \mathbf{D} \rangle$ and $\langle \mathbf{A} \vee \mathbf{D} \vee \mathbf{E} \rangle$ (Fig. 7). Therefore, thinned code is bound—representation of 1s from each component depends on the other components present.

Similar component sets produce similar thinned codes, preserving structured similarity. However, in contrast to the mentioned binding schemes, the character of structured similarity here is not multiplicative in terms of component codevectors. In those schemes, a single dissimilar component makes the binding dissimilar. For this procedure, due to the initial superposition, a single similar component makes the resulting thinned codevectors similar.

The character of structured similarity is shown in Fig. 8. The input vector was the superposition of five independent component codevectors of the same density. The similarity of two input vectors was varied by changing the composition of components. For example, if we choose the first codevector as $\mathbf{Z1} = \mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \vee \mathbf{D} \vee \mathbf{E}$ and the second one as $\mathbf{Z2} = \mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \vee \mathbf{F} \vee \mathbf{H}$, their similarity (overlap) will be approximately 0.6. Depending on the density of the thinned codevectors, their similarity varies from linear

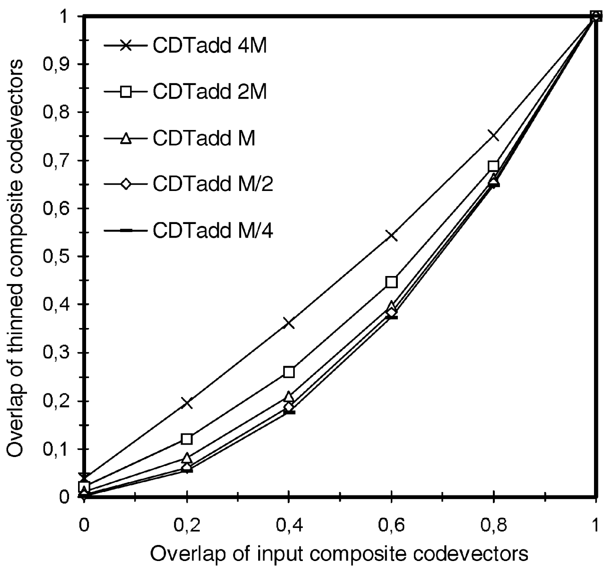

Fig. 8. Overlap of thinned composite codevectors for various "depth" of thinning. There are five components in the composite item. For all component codevectors, $N = 100,000$ and $M \approx 1,000$. Therefore, the input composite codevector includes about $5\mathrm{M}$ of 1s. The resulting codevectors were thinned to have from $4\mathrm{M}$ to $\mathrm{M}/4$ of 1s.

(additive similarity of superposition corresponding to the "shallow" thinning) to approximately quadratic (similarity of about 0.55 for $4M$ versus similarity of about 0.37 for $M/4$).

2.4 Unbinding

To retrieve component codevectors from their bindings, various techniques are used in different representation schemes.

A special unbinding operation is needed for HRRs and BSCs because their bindings are not similar to the codevectors of bound components. To reconstruct $\mathbf{Y}$ from $\mathbf{Z} = \mathbf{X} * \mathbf{Y}$, $\mathbf{X}$ should be known. If $\mathbf{X}$ is known, its inverse can be found. According to Plate (e.g., [40], [41]), the approximate inverse of $\mathbf{X} : \mathbf{X}^{\mathrm{T}}$ is enough for unbinding and easy to obtain: $X_{\mathrm{i}}^{\mathrm{T}} = X_{\mathrm{N-i}}$, where subscripts are modulo-N. Unbinding gives

$$\mathbf{X}^{\mathrm{T}} * \mathbf{Z} = \mathbf{X}^{\mathrm{T}} * \mathbf{X} * \mathbf{Y} = \mathbf{Y} + \text{ noise}. \tag{7}$$

To reconstruct an accurate version of $\mathbf{Y}$, the noisy version can be compared with all component codevectors to find the closest one.

In BSCs, unbinding is implemented by XOR.

In APNNs, the similarity of component codevectors to their thinned composite codevectors is preserved. Therefore, in order to retrieve in full the component codevectors using the thinned codevectors, no intermediate unbinding operation is necessary—it is enough to find the most similar codevectors in the component memory. Memory organization and the decoding of hierarchical APNN structures is discussed in Section 4.1.

2.5 Normalization

The operation of normalization is used in HRRs to maintain the overall strength of codevectors and the statistical distribution of their elements after other operations have been performed on them. In HRRs, the normalized codevector $\mathbf{X}$ is denoted as $\langle\mathbf{X}\rangle$ and is calculated as $\langle\mathbf{X}\rangle = \mathbf{X}/\mid\mathbf{X}\mid$, where $\mid\mathbf{X}\mid$ is the Euclidean length of $\mathbf{X}$.

In BSCs, normalization is implemented by thresholding each element of the superposition codevector. This maintains the binary character of the resulting codevector and its density of 0.5.

In APNNs, the CDT procedure not only binds the codevectors of components, but also implements normalization by controlling the density of the resulting codevectors.

3 Distributed Representation of Relational Structures

Let us consider how operations and procedures discussed above can be used to represent relational structures.

3.1 HRRs and BSCs for Relational Structures

HRRs and BSCs both use role-filler bindings to represent relational structure (e.g., [42], [26]). The relational instance or proposition "Spot bit Jane" is represented by HRRs as:

$$\mathbf{P}(\mathbf{bite}) = \langle\mathbf{P\_structure}(\mathbf{bite}) + \mathbf{P\_components}(\mathbf{bite})\rangle, \tag{8}$$

where

$$\mathbf{P\_structure}(\mathbf{bite}) = \mathbf{bite\_agt} * \mathbf{spot} + \mathbf{bite\_obj} * \mathbf{jane}, \tag{9}$$

$$\mathbf{P\_components}(\mathbf{bite}) = \mathbf{bite} + \langle\mathbf{spot} + \mathbf{jane}\rangle. \tag{10}$$

$\mathbf{P\_structure}(\mathbf{bite})$ represents the order entities Spot and Jane possess in their "bite" relationship. The codevectors **spot** and **jane** are fillers bound correspondingly to the codevectors **bite_agt** and **bite_obj**, which represent biting agent and bitten object, respectively.

$\mathbf{P\_components}(\mathbf{bite})$ represents the entities involved and relation label **bite** (the latter could be replaced by $\mathbf{bite\_agt} + \mathbf{bite\_obj}$). It preserves the unstructured similarity of $\mathbf{P}(\mathbf{bite})$ to its component codevectors (which is not preserved by bindings in $\mathbf{P\_structure}(\mathbf{bite})$). It is needed both to support the surface (superficial) similarity of representation to its constituents and to be able to reconstruct by unbinding which filler is bound with which role. For example, unstructured similarity allows for the finding of **bite_agt**, **bite_obj**, **spot**, and **jane** in $\mathbf{P}(\mathbf{bite})$ and then the corresponding fillers or roles:

$$\mathbf{bite\_agt}^{\mathrm{T}} * \mathbf{P}(\mathbf{bite}) = \mathbf{spot} + \text{noise}, \tag{11}$$

or

$$\mathbf{jane}^{\mathrm{T}} * \mathbf{P}(\mathbf{bite}) = \mathbf{bite\_obj} + \text{noise}. \tag{12}$$

In HRRs, to represent higher-level propositions which use lower-level propositions as their components, the lower-level propositions are used as the fillers for some higher-level roles. For example, the representation of "Spot bit Jane, causing Jane to flee from Spot" looks as [42]:

$$\mathbf{P} = \langle \mathbf{cause} + \langle \mathbf{P}(\mathbf{bite}) + \mathbf{P}(\mathbf{flee})\rangle + \mathbf{cause\_antc} * \mathbf{P}(\mathbf{bite}) + \mathbf{cause\_cnsq} * \mathbf{P}(\mathbf{flee})\rangle, \tag{13}$$

where $\mathbf{P}(\mathbf{bite})$ is as above and $\mathbf{P}(\mathbf{flee})$ is formed analogously:

$$\mathbf{P}(\mathbf{flee}) = \langle \mathbf{flee} + \langle \mathbf{spot} + \mathbf{jane}\rangle + \mathbf{flee\_agt} * \mathbf{jane} + \mathbf{flee\_obj} * \mathbf{spot}\rangle. \tag{14}$$

3.2 Representation of Relational Structure in APNNs

Representation of relational structure in APNNs has some peculiarities compared to the HRR representations considered above. Two types of binding may be distinguished in the APNN representation. The first type is analogous to the role-filler bindings in HRRs and is used to represent the correspondence or order of items. The second type is used to bind together a set of component items comprising a composite item. This type of binding is not explicitly used in HRRs.

3.2.1 Binding for Representation of Correspondence or Order

One version of encoding the correspondence of a filler to a certain role is analogous to the role-filler bindings in HRRs. In APNNs, it is realized by the explicit binding of the role codevector with the filler codevector. If the role is just a "position" (1st, 2nd, 3rd, ...) of an item in a proposition, this scheme encodes the order of filler items (see also [45], [32], [46]). The binding can be implemented by some version of the CDT or hetero-CDT procedure. (Hetero-thinning is a special kind of thinning where the codevectors to be thinned are not superimposed [45], [32], [46]).

Role-filler bindings using the CDT procedure are exemplified by ${}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle$, ${}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle$, where $\mathbf{bite\_agt}$ corresponds to the "1st place role in *bite* proposition" and $\mathbf{bite\_obj}$ corresponds to the "2nd place role in *bite* proposition." ${}^1\langle \ldots\rangle$ denotes thinning using the same thinning pattern, as introduced in Section 2.3.3.

Another version of order encoding consists in changing the physical position of an item codevector depending on its position (or place or ordinal number) in a sequence or proposition. In APNNs, it is implemented by a (cyclic) shift [31] or other invertible permutation of the item codevector. Similar encoding is considered in [49] and [12].

For our example, it can be represented as

$$\mathbf{spot} \gg \mathrm{shift\_agent}, \mathbf{jane} \gg \mathrm{shift\_object}.$$

Here, $\mathbf{Z} \gg \mathrm{q}$ is the shift of $\mathbf{Z}$ by q bits, $\mathrm{shift\_agent}$ and $\mathrm{shift\_object}$ are the numbers of one-bit shifts chosen for the agent and object roles. These shifts (or permutations) should not coincide with those used for thinning (see Section 2.3.3, and Figs. 5 and 6). This type of order representation may be related to symbol-argument-argument (or predicate-arguments) representation (e.g., [17]).

3.2.2 Binding Together a Set of Items

As exemplified in Section 3.1, in the HRR scheme a proposition is represented by superimposed codevectors of component subpropositions and their bindings. In its turn, each subproposition is a superposition of its own components. It means that in HRRs, propositions of all composition levels are represented by a superposition of codevectors which are not bound together.

Unbound representations of propositions, when superimposed, can be confused. For example, the propositions "Spot bit Jane, Fido bit Fred" and "Spot bit Fred, Fido bit Jane" will produce indistinguishable codevectors:

$$\mathbf{P}(\mathbf{bite}, \mathbf{spot\text{-}jane}) + \mathbf{P}(\mathbf{bite}, \mathbf{fido\text{-}fred}) = \mathbf{P}(\mathbf{bite}, \mathbf{spot\text{-}fred}) + \mathbf{P}(\mathbf{bite}, \mathbf{fido\text{-}jane}). \tag{15}$$

This can be seen from the proper superposition of their $\mathbf{P_structure}$s:

$$\begin{aligned}
&\mathbf{P\_structure}(\mathbf{bite}, \mathbf{spot\text{-}jane}) \\
&\quad = \mathbf{bite\_agt} * \mathbf{spot} + \mathbf{bite\_obj} * \mathbf{jane}, \\
&\mathbf{P\_structure}(\mathbf{bite}, \mathbf{fido\text{-}fred}) \\
&\quad = \mathbf{bite\_agt} * \mathbf{fido} + \mathbf{bite\_obj} * \mathbf{fred}, \\
&\mathbf{P\_structure}(\mathbf{bite}, \mathbf{spot\text{-}fred}) \\
&\quad = \mathbf{bite\_agt} * \mathbf{spot} + \mathbf{bite\_obj} * \mathbf{fred}, \\
&\mathbf{P\_structure}(\mathbf{bite}, \mathbf{fido\text{-}jane}) \\
&\quad = \mathbf{bite\_agt} * \mathbf{fido} + \mathbf{bite\_obj} * \mathbf{jane}.
\end{aligned} \tag{16}$$

The same is true for $\mathbf{P\_components}$.

This is a kind of superposition catastrophe mentioned in Section 2.3.1. (Some relevant issues are also discussed in [39], [17], [12]). To avoid it, such a representation scheme requires a role in the higher-level proposition for binding with each proposition of the lower level. This limits the flexibility of structure representation. This also increases the risk of "spurious memories" in a distributed memory with a superposition learning rule if it is used to store unbound propositions (see Section 4).

In contrast to HRRs, superpositions of codevectors in APNNs are typically subjected to binding. In particular, all codevectors superimposed in a proposition can be bound together by the CDT procedure. For example, the propositions of (16) will look as follows:

$$\begin{aligned}
&\mathbf{P\_apnn\text{-}structure}(\mathbf{bite}, \mathbf{spot\text{-}jane}) \\
&\quad = {}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle, \\
&\mathbf{P\_apnn\text{-}structure}(\mathbf{bite}, \mathbf{fido\text{-}fred}) \\
&\quad = {}^2\langle {}^1\mathbf{bite\_agt} \vee \mathbf{fido}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{fred}\rangle\rangle, \\
&\mathbf{P\_apnn\text{-}structure}(\mathbf{bite}, \mathbf{spot\text{-}fred}) \\
&\quad = {}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{fred}\rangle\rangle, \\
&\mathbf{P\_apnn\text{-}structure}(\mathbf{bite}, \mathbf{fido\text{-}jane}) \\
&\quad = {}^2\langle\ {}^1\langle \mathbf{bite\_agt} \vee \mathbf{fido}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle.
\end{aligned} \tag{17}$$

Then, the codevectors of propositions "Spot bit Jane, Fido bit Fred" and "Spot bit Fred, Fido bit Jane" will no longer be the same:

$$\begin{aligned}
&{}^3\langle \mathbf{P\_apnn}(\mathbf{bite}, \mathbf{spot\text{-}jane}) \vee \mathbf{P\_apnn}(\mathbf{bite}, \mathbf{fido\text{-}fred})\rangle \neq \\
&{}^3\langle \mathbf{P\_apnn}(\mathbf{bite}, \mathbf{spot\text{-}fred}) \vee \mathbf{P\_apnn}(\mathbf{bite}, \mathbf{fido\text{-}jane})\rangle.
\end{aligned} \tag{18}$$

Let us note that if each proposition of (16) is normalized, the equality in (15) will not be exact. However, normalization in HRRs has not been considered as a binding operation.

3.2.3 Examples of Relational Structure Representation

Let us consider the peculiarities of possible APNN representations of relational structures from Section 3.1.

Using role-filler bindings for order representation, $\mathbf{P\_structure}(\mathbf{bite})$ of (9) can be represented as in (17):

$$\begin{aligned}&\mathbf{P\_apnn\text{-}structure}(\mathbf{bite}, \mathbf{spot\text{-}jane})\\&\quad = {}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle.\end{aligned} \tag{19}$$

However, this representation need not be augmented with $\mathbf{P\_components}(\mathbf{bite})$ as in (8). Since the CDT procedure preserves unstructured similarity, ${}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle$ is already similar to its components. Also, it is possible to decode (19) through its components ${}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle$ and ${}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle$, and then to decode the latter through their roles and fillers using unstructured similarity (see Section 4.1).

In the same manner, representation of $\mathbf{P}(\mathbf{flee})$ (14) and $\mathbf{P}$ (13) can be simplified using APNN representations with role-filler bindings:

$$\begin{aligned}\mathbf{P\_apnn}(\mathbf{bite}) = {}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle\\ \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle,\end{aligned} \tag{20}$$

$$\begin{aligned}\mathbf{P\_apnn}(\mathbf{flee}) = {}^2\langle {}^1\langle \mathbf{flee\_agt} \vee \mathbf{jane}\rangle\\ \vee {}^2\langle \mathbf{flee\_obj} \vee \mathbf{spot}\rangle\rangle,\end{aligned} \tag{21}$$

$$\begin{aligned}\mathbf{P\_apnn} = {}^4\langle {}^3\langle \mathbf{cause\_antc} \vee \mathbf{P\_apnn}(\mathbf{bite})\rangle\\ \vee {}^3\langle \mathbf{cause\_cnsq} \vee \mathbf{P\_apnn}(\mathbf{flee})\rangle\rangle.\end{aligned} \tag{22}$$

Using a shift (or an other invertible permutation) of codevectors to encode the order, representation can be as follows:

$$\begin{aligned}&\mathbf{P\_apnn1\text{-}structure}(\mathbf{spot\text{-}jane})\\&\quad = \mathbf{spot} \gg \mathrm{agent} \vee \mathbf{jane} \gg \mathrm{object}.\end{aligned} \tag{23}$$

However, since such a representation is not similar at all to

$$\begin{aligned}&\mathbf{P\_apnn1\text{-}structure}(\mathbf{jane\text{-}spot})\\&\quad = \mathbf{jane} \gg \mathrm{agent} \vee \mathbf{spot} \gg \mathrm{object},\end{aligned}$$

it should be augmented with the component representation, in the same manner as for HRRs:

$$\begin{aligned}&\mathbf{P\_apnn1\text{-}components}(\mathbf{bite}, \mathbf{spot}, \mathbf{jane})\\&\quad = \mathbf{bite} \vee \mathbf{spot} \vee \mathbf{jane}.\end{aligned} \tag{24}$$

So, the proposition $bite(spot, jane)$ will look like:

$$\begin{aligned}&\mathbf{P\_apnn1}(\mathbf{bite}) ={}^1\langle \mathbf{P\_apnn1\text{-}structure}(\mathbf{spot\text{-}jane})\\&\quad \vee \mathbf{P\_apnn1\text{-}components}(\mathbf{bite}, \mathbf{spot}, \mathbf{jane})\rangle.\end{aligned} \tag{25}$$

$\mathbf{P\_apnn1}(\mathbf{flee})$ is constructed in the same manner:

$$\begin{aligned}&\mathbf{P\_apnn1}(\mathbf{flee}) = {}^1\langle \mathbf{P\_apnn1\text{-}structure}(\mathbf{jane\text{-}spot})\\&\quad \vee \mathbf{P\_apnn1\text{-}components}(\mathbf{flee}, \mathbf{jane}, \mathbf{spot})\rangle,\end{aligned} \tag{26}$$

and the whole proposition looks like:

$$\begin{aligned}&\mathbf{P\_apnn1} = {}^2\langle \mathbf{cause} \vee \mathbf{P\_apnn1}(\mathbf{bite})\\&\quad \vee \mathbf{P\_apnn1}(\mathbf{flee}) \vee \mathbf{P\_apnn1}(\mathbf{bite}) \gg \mathrm{cause\_antc}\\&\quad \vee \mathbf{P\_apnn1}(\mathbf{flee}) \gg \mathrm{cause\_cnsq}\rangle.\end{aligned} \tag{27}$$

As another example, let us show a possible APNN-style representation for labeled directed acyclic graph $\phi(\alpha, \psi(\gamma), \psi(\gamma, \phi(\alpha, \beta)))$ taken from [10]:

$$\begin{aligned}&{}^3\langle \phi \vee \alpha \gg 1 \vee {}^2\langle \psi \vee \gamma \gg 1\rangle \gg 2 \vee {}^2\langle \psi \vee \gamma \gg 1\\&\quad \vee {}^1\,\langle \phi \vee \alpha \gg 1 \vee \beta \gg 2\rangle \gg 2\rangle \gg 3\rangle.\end{aligned}$$

4 Memory

4.1 Memory Organization

Code vectors of items of various composition levels should be stored in memory. It allows finding the closest match for a codevector. It also allows finding the components of a composite item.

In HRRs and in BSCs, fillers of all nesting levels are considered "chunks" and should be stored in memory. For all nesting levels (except may be the lowest one), those chunks or fillers are superpositions of codevectors and bindings of the previous level(s).

Code vectors of chunks and subchunks of all composition or nesting levels are stored in one memory array (large-scale clean-up memory [42]). They are used in the reconstruction of components of complex structures. For example, to decode $\mathbf{P}$ (13), first its component codevector $\mathbf{cause}$ (or $\mathbf{cause\_antc}$ and $\mathbf{cause\_cnsq}$) should be retrieved using unstructured similarity. Then, to decode, e.g., the cause antecedent, $\mathbf{P}$ should be unbound using $\mathbf{cause\_antc}^{\mathrm{T}}$. The resulting noisy version of $\mathbf{P}(\mathbf{bite})$ can be cleaned up by accessing the memory and retrieving the closest match. In turn, the reconstructed $\mathbf{P}(\mathbf{bite})$ can be decoded through the lower level components (as in (11) and (12)).

In APNNs, the memory is organized hierarchically [29], [33]. Code vectors corresponding to composite items of different composition levels are stored in different memory arrays. Accordingly, codevectors of the same composition level are stored in the same memory array. The memory at each level should perform retrieval of the closest match(es) to a probe codevector.

Such an organization of memory allows a simple traversing of compositional hierarchy using unstructured similarity. If we construct a probe codevector of some intermediate hierarchical level, we can find a similar codevector of the same compositional complexity in the memory array of the same compositional level. We can find the components of the probe by probing the memory arrays of the lower level(s). We can also find more complex codevectors that contain the probe as their component by probing the higher level(s) of memory. If all codevectors were in the same memory array, one could not determine solely by their unstructured similarity to the probe whether a superset or a subset codevector is found—consideration of

TABLE 1
The Memory Organization for the Role-Filler APNN-Style Representation of the Nested Proposition "Spot Bit Jane, Causing Jan to flee from Spot" for (20), (21) and (22)

Composition or memory level	Codevectors of items or chunks	Configuration of thinning
level#4	$\mathbf{Probe} = {}^4\langle\, \mathbf{Antc} \vee \mathbf{Cnsq}\,\rangle$	${}^4\langle\ \rangle$
level#3	$\mathbf{Antc} = {}^3\langle\, \mathbf{cause\_antc} \vee \mathbf{bite}\,\rangle$ $\mathbf{Cnsq} = {}^3\langle\, \mathbf{cause\_cnsq} \vee \mathbf{flee}\,\rangle$	${}^3\langle\ \rangle$
level#2	$\mathbf{Bite} = {}^2\langle {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\,\rangle \vee {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle\rangle$ $\mathbf{Flee} = {}^2\langle {}^1\langle \mathbf{flee\_agt} \vee \mathbf{jane}\rangle \vee {}^1\langle \mathbf{flee\_obj} \vee \mathbf{spot}\rangle\rangle$	${}^2\langle\ \rangle$
level#1	$\mathbf{bite\_a} = {}^1\langle \mathbf{bite\_agt} \vee \mathbf{spot}\rangle$ $\quad$ $\mathbf{bite\_o} = {}^1\langle \mathbf{bite\_obj} \vee \mathbf{jane}\rangle$ $\mathbf{flee\_a} = {}^1\langle \mathbf{flee\_agt} \vee \mathbf{jane}\rangle$ $\quad$ $\mathbf{flee\_o} = {}^1\langle \mathbf{flee\_obj} \vee \mathbf{spot}\rangle$	${}^1\langle\ \rangle$
level#0	$\mathbf{spot} = \mathbf{dog} \vee \mathbf{id\_spot}$ $\quad$ $\mathbf{jane} = \mathbf{human} \vee \mathbf{id\_jane}$ **bite_agt** **bite_obj** **flee_agt** **flee_obj** **cause_antc** **cause_cnsq**	
base-level	**human** **dog** **cat** **mouse** **id_jane** **id_john** **id_fred** **id_felix** **id_fido** **id_spot** **id_rover** **id_mort** **bite_agt** **bite_obj** **flee_agt** **flee_obj** **cause_antc** **cause_cnsq**	

The codevector of each level of composition is stored in a memory array corresponding to that level. Code vectors of composite items (level #1 - level #4) are thinned to 2M. *Noncomposite elements of level #1 coincide with the base-level codevectors of level #0.*

the upper or the lower level roles would be required, as in the role-filler scheme of HRRs.

For the role-filler representation of (20), (21), and (22), $\mathbf{spot}$, $\mathbf{jane}$, $\mathbf{bite\_agt}$, $\mathbf{bite\_obj}$, $\mathbf{flee\_agt}$, $\mathbf{flee\_obj}$, $\mathbf{cause\_antc}$, and $\mathbf{cause\_cnsq}$ (composition level #0) are stored in memory level #0 (Table 1). Bindings ${}^1\langle \mathbf{spot} \vee \mathbf{bite\_agt}\rangle$, ${}^1\langle \mathbf{jane} \vee \mathbf{bite\_obj}\rangle$, ${}^1\langle \mathbf{jane} \vee \mathbf{flee\_agt}\rangle$, and ${}^1\langle \mathbf{spot} \vee \mathbf{flee\_obj}\rangle$ are stored in memory level #1. Code vectors $\mathbf{P\_apnn}(\mathbf{bite})$ of (20) (denoted simply as $\mathbf{Bite}$ in Table 1) and $\mathbf{P\_apnn}(\mathbf{flee})$ of (21) (denoted as $\mathbf{Flee}$) are stored in memory level #2. Code vectors ${}^3\langle \mathbf{cause\_antc} \vee \mathbf{bite}\rangle = \mathbf{Antc}$ and ${}^3\langle \mathbf{cause\_cnsq} \vee \mathbf{flee}\rangle = \mathbf{Cnsq}$ are stored in memory level #3. Codevector $\mathbf{P} = \mathbf{Probe}$ is stored in memory level #4.

To find a similar codevector in memory, the probe codevector $\mathbf{P}$ should be compared with other available codevectors of its memory level. For example, $\mathbf{P}$ should be primarily compared with other codevectors of level #4. (Such codevectors will appear for different analogous episodes in Section 5.1.3).

Retrieval of the components of a composite item is done recursively. First, its codevector should be compared with the codevectors of the immediately lower memory level (in other versions, with several lower levels). For example, $\mathbf{P}$ should be compared with the codevectors of level #3 to find its component bindings $\langle \mathbf{cause\_antc} \vee \mathbf{P\_bite}\rangle$ and $\langle \mathbf{cause\_cnsq} \vee \mathbf{P\_flee}\rangle$. In turn, they could be decoded through the lower level components, etc. When we reach the codevector $\langle \mathbf{spot} \vee \mathbf{bite\_agt}\rangle$ of memory level #1, it should be compared with the codevectors of memory level #0 to find its component codevectors $\mathbf{spot}$ and $\mathbf{bite\_agt}$. For decoding, it is not needed to compare a codevector of memory level #1 with the codevectors of higher memory levels.

If the task is to decode which filler is bound to a specific role (without full decoding of a proposition), the role is first used as a probe to recover its binding (as the closest match) and, then the binding is decoded to discover the filler. For example, to discover the filler of the $\mathbf{bite\_agt}$ role, it is used as a probe to retrieve the ${}^1\langle \mathbf{spot} \vee \mathbf{bite\_agt}\rangle$ binding of memory level #1 and, then the binding is used as a probe of memory level #0 to discover $\mathbf{spot}$ as the closest match. (It is possible to eliminate $\mathbf{bite\_agt}$ representation from the probe: ${}^1\langle \mathbf{spot} \vee \mathbf{bite\_agt}\rangle \wedge \neg \mathbf{bite\_agt}$ in order to omit $\mathbf{bite\_agt}$ as the other closest match).

Possible hierarchical levels for the shift-binding representation of (23), (24), (25), (26), and (27) are shown in Table 2. Decoding should include the inverse of shifts (or other permutations) used to encode the agent-object roles.

4.2 Memory Implementation

As mentioned in the previous sections, memory in HRRs, BSCs, and APNNs should be able to return the closest vector(s) to the input (performing error-correcting retrieval).

Since in HRRs the chunks are superpositions of codevectors not bound together, the memory where they are stored should not be a distributed memory with a superposition learning rule in order to avoid the risk of superposition catastrophe and "spurious memories" (see Section 2.3.1 and [46]). Plate [42] considers, as appropriate, the auto-associative memory schemes of Baum et al. [3]. In the simplest implementation of "grandmother cell" memory, the memory is a neural network that contains a number of hidden units. Each unit is devoted to a single stored codevector. The input and output weights of a hidden unit are the reference codevector that the unit stores. During retrieval, the hidden unit closest to the noisy input codevector becomes active and outputs the cleaned version of that codevector.

In APNNs, the codevectors of all chunks are processed by the CDT procedure, which means they are bound and normalized (see Sections 2.3 and 2.5).

Since the codevectors of chunks are bound, the risk of spurious memories is reduced and distributed memories

TABLE 2
The Memory Organization for the Shift-Binding APNN-Style Representation of the Nested Proposition "Spot Bit Jane, Causing Jane to Flee from Spot" for (23), (24), (25), (26), and (27)

Composition or memory level	Codevectors of items or chunks	Configuration of thinning
level#2	**P** = ${}^2\langle$**cause** $\vee$ **BITE** $\vee$ **FLEE** $\vee$ **BITE**>>cause_antc $\vee$ **FLEE**>>cause_cnsq$\rangle$	${}^2\langle\ \rangle$
level#1	**BITE** = ${}^1\langle$**bite** $\vee$ **spot** $\vee$ **jane** $\vee$ **spot**>>agent $\vee$ **jane**>>object$\rangle$ **FLEE** = ${}^1\langle$**flee** $\vee$ **spot** $\vee$ **jane** **jane**>>agent $\vee$ **spot**>>object$\rangle$	${}^1\langle\ \rangle$
level#0	**spot = dog** $\vee$ **id_spot** **jane = human** $\vee$ **id_jane** **bite** **flee** **cause**	
base-level	**human** **dog** **cat** **mouse** **id_jane** **id_john** **id_fred** **id_felix** **id_fido** **id_spot** **id_rover** **id_mort** **bite** **flee** **cause**	

The codevector of each level of composition is stored in a memory array corresponding to that level. Code vectors of composite items (level #1, level #2) are thinned to 4M.

with simple (generic and biologically relevant) superposition learning rules, such as versions of Hebb's rule [19], are allowed at each hierarchical level.

Since the codevectors of chunks are normalized, the chunks of all levels are sparse binary codevectors. It is well-known that matrix-style distributed memories offer a high capacity (in terms of the number of stored codevectors) and a good error-correcting ability for sparse codevectors (e.g., [59], [37], [34], [1], [11], [54]). Though the maximal capacity is reached at $M \approx \log N$, we used the larger M for statistical stability. Even for $N^{1/3} < M < N^{1/2}$, the number of stored codevectors that can be reconstructed from their noisy versions can significantly exceed N (e.g., [44], [45]).

Besides, computational implementation of storage and retrieval operations (vector outer-product or vector-matrix multiplication) becomes very simple and fast for binary codevectors because it does not require at all floating point operations and even multiplication. Further acceleration is possible for sparse codevectors (e.g., [38], [33]). This is all true for the operations of finding codevector overlap, superposition, and binding, which basically reduce to bit operations on codevectors.

These considerations are also true for Kanerva's Sparse Distributed Memory, which is also a candidate for long-term memory implementation [25], [50], [42].

Thus, each of the memory arrays mentioned in Section 4.1 could be an auto-associative memory of the type discussed in this section. The issues concerning pros and cons of their possible implementation in the same physical memory should be further investigated. However, at the current stage of modeling, when not too many items are used, functioning of a memory level can be simulated by storing codevectors of that level in a list and performing an exhaustive comparison for the closest match. In particular, we use such a technique in the experiments of the following sections.

5 Modeling of Analogical Reasoning

Analogical thinking is one of the most commonly encountered cognitive processes associated with the manipulation of structured information. That is why a lot of work is devoted to its study and modeling. (For an introduction, see, e.g., [14], [15], [53], [24] and references therein).

Analogs are matched not only by "surface" or "superficial similarity" (see, e.g., [13], [9]) based on common elements (attributes, objects, relations, propositions) or a broader "semantic similarity" (see, e.g., [53], [6]) of those elements. Of no less importance is structural similarity which is determined by how the elements of analogs are arranged with respect to each other. This is based on the notion of "structural consistency" [14], [7] or "isomorphism" [53], [24]. These psychological constraints on structural similarity require one-to-one correspondence (at most one element in one analog corresponds to one element in the other) and parallel connectivity (corresponding relations must have corresponding arguments).

The first two stages in analogical processing are access and mapping. Access (retrieval, recall, recognition) is the process of finding, in memory, the most appropriate source (base) analog(s) given a target (probe, cue) situation. Mapping is the process of finding correspondences between the elements of two analogs.

5.1 Analogical Retrieval

Though there are different views on the relative role of surface and structural similarity in analogical retrieval, the models of access, which take into account only surface similarity, are considered inadequate.

5.1.1 *An Example of the Retrieval Task*

To illustrate analogical access by estimating the similarity of distributed representations, the episodes adapted by Plate [42] from [53] will be used.

The following characters or entities are involved: dogs (Fido, Spot, Rover), people (Jane, John, Fred), a cat (Felix),

TABLE 3
Similarity Scores Between the Code Vectors Representing the Probe and the Episodes

Type of similarity	Episode	Similarity scores: HRR [42], role-filler	APNN, role-filler	APNN, shift-binding
P	Spot bit Jane, causing Jane to flee from Spot	1.00	1.00	1.00
LS	Fido bit John, causing John to flee from Fido	0.71	0.42	0.40
SF	John fled from Fido, causing Fido to bite John	0.47	0.38	0.24
CM	Fred bit Rover, causing Rover to flee from Fred	0.47	0.38	0.30
AN	Mort bit Felix, causing Felix to flee from Mort	0.42	0.26	0.14
FOR	Mort fled from Felix, causing Felix to bite Mort	0.30	0.25	0.09
	Standard deviation range	0.016-0.026	0.009-0.013	0.005-0.008

HRR scores are given in the first column. The scores for two schemes of APNN-style representation (role-filler, (20), (21), and (22), and shift-binding, (23), (24), (25), (26), and (27)) are presented in the second and in the third columns. The results are averaged over 100 runs with different random base-level codevectors.

and a mouse (Mort). Relations are bite, flee, and cause. The "probe" episode P, which is compared to the others, is the one used previously as an example for structure representation: "Spot bit Jane, causing Jane to flee from Spot." Other episodes have the same relations as the probe but different types of similarity—mainly according to Gentner's classification ([13], see also, e.g., [9]).

A literal similarity (LS) episode has both structural and superficial similarity to the probe: "Fido bit John, causing John to flee from Fido." A surface features (SF) episode has superficial but not structural similarity: "John fled from Fido, causing Fido to bite John." A cross-mapped analogy (CM) episode has both structural and superficial similarity, but the types of corresponding entities are switched: "Fred bit Rover, causing Rover to flee from Fred." An analogy (AN) episode has structural but not superficial similarity: "Mort bit Felix, causing Felix to flee from Mort." A first order relations only (FOR) episode has neither structural nor superficial similarity, other than shared predicates: "Mort fled from Felix, causing Felix to bite Mort."

Human experimental data generally support the notion that surface similarity is more important for analogical retrieval than structural similarity (experimental results are reported, e.g., in [9], [58], [47]). In terms of analogical similarity types, it means that analogs of similarity type LS, CM, and SF are easier to retrieve than those of similarity type AN and FOR. Among the group of situations with surface similarity, LS is more easily accessible than CM and SF. The difference between CM and SF is not clear.

Analyzing data from various authors, Plate [42] provides the following pattern for retrievability of analogs from long-term memory:

$$\mathrm{LS} > \mathrm{CM} \geq \mathrm{SF} > \mathrm{AN} \geq \mathrm{FOR}. \qquad (28)$$

5.1.2 Estimating Analogical Similarity with HRRs

The scheme for episode encoding was considered in Section 3.1. To build HRRs according to that scheme, it is necessary to encode the entities and relations using base-level codevectors. By base-level codevectors, we mean here independent codevectors of the lowest composition level (level #0). As mentioned in Section 2.1, in HRRs, the codevectors of independent items are N dimensional real-valued vectors with the elements chosen randomly and independently using the Gaussian distribution with 0 mean and variance $1/N$. After generation, they are fixed. For the experiments reported in [42], Plate used codevectors of $N = 2,048$.

All relations and roles were considered to be dissimilar and were represented by the base-level codevectors, e.g., **bite**, **bite_agt**, **bite_obj**, etc. The codevectors of entities (or tokens of some type) were formed as the superposition of base-level codevectors of two features: that of type (**human**, **dog**, **cat**, **mouse**) and that of identifier or name (**id_john**, **id_fido**, etc.). For example, the entities John and Mort were represented, respectively, as $\mathbf{john} = \langle \mathbf{human} + \mathbf{id\_john} \rangle$ and $\mathbf{mort} = \langle \mathbf{mouse} + \mathbf{id\_mort} \rangle$.

The similarity scores obtained by Plate [42] are shown in the first column of Table 3. These results show that the similarity of HRRs encoding the episodes is influenced both by their superficial and structural similarity. HRRs are similar if they include similar relations or entities (episodes AN, FOR). The similarity is higher if they include both similar relations and entities (episodes CM, SF). The similarity is still higher if similar entities fill similar roles in similar relations (episode LS). These results correspond to the experimental data expressed by (28): $\mathrm{LS} > \mathrm{CM} \geq \mathrm{SF} > \mathrm{AN} \geq \mathrm{FOR}$.

5.1.3 Estimating Analogical Similarity with APNNs

We have conducted experiments in analogical similarity estimation using APNN representations. Random binary vectors with $N = 100,000$ and $M = 1,000$ were used as the base-level codevectors. Entities were presented as the superpositions of base-level codevectors corresponding to the same features as in HRRs of Section 5.1.2 (see also base-level and level #0 of Table 1). Both schemes of Section 3.2.3: the role-filler scheme of (20), (21), and (22), and the shift-binding scheme of (23), (24), (25), (26), and (27) were used to construct the representations of episodes.

For the role-filler scheme of (20), (21), and (22), the propositions of level #1 and higher levels (as in Table 1) were thinned down to $2\mathrm{M}$. This depth of thinning was chosen empirically to obtain the similarity scores between the codevector of the probe and that of each episode presented in the second column of Table 3. Other depths of thinning could change the score pattern; however,

TABLE 4
The Similarity Matrix for the Elements of the Probe and the LS Episode

		level#4	level#3		level#2		level#1			
		Probe	**Antc**	**Cnsq**	**Bite**	**Flee**	**bite_a**	**bite_o**	**flee_a**	**flee_o**
level#4	**Epis(LS)**	0.41	0.30	0.31	0.20	0.20	0.15	0.15	0.15	0.14
level#3	**Antc**	0.30	0.50	0.09	0.29	0.12	0.21	0.21	0.11	0.10
	Cnsq	0.30	0.09	0.51	0.12	0.30	0.10	0.11	0.20	0.20
level#2	**Bite**	0.19	0.28	0.11	0.41	0.16	0.29	0.29	0.15	0.13
	Flee	0.19	0.12	0.28	0.17	0.43	0.14	0.14	0.29	0.29
level#1	**bite_a**	0.14	0.20	0.09	0.28	0.13	0.56	0.02	0.02	0.24
	bite_o	0.14	0.21	0.10	0.28	0.14	0.03	0.55	0.26	0.03
	flee_a	0.15	0.11	0.20	0.14	0.30	0.02	0.26	0.57	0.02
	flee_o	0.15	0.10	0.20	0.13	0.29	0.25	0.02	0.02	0.56

Notations are the same as in Table 1.

discussion of this influence is beyond the scope of the present paper.

For the shift-binding scheme of (23), (24), (25), (26), and (27), the propositions of level #1 and level #2 (as in Table 2) were thinned to $4M$. The similarity scores for this depth of thinning are given in the third column of Table 3. Note that thinning is rather shallow at level #1: from approximately $5M$ to $4M$, but at level #2 it is rather deep: from approximately $15M$ to $4M$. ($15M$ instead of $4M * 4 + M = 17M$ is obtained because of the "absorption" of coincident 1s in codevectors during superposition).

The order of similarity scores for both types of APNN representations considered is the same as for HRRs. It is also consistent with the pattern of experimental results observed for people ([9], [58], [47]) and modeling results reported for MAC/FAC [9] and ARCS [53] (where the data different from those in this paper were used).

5.2 Mapping of Analogies

The second step in analogy processing after the retrieval of an analogous episode is usually mapping or the interpretation of the analogy. When both analogs are known, the task is to find the corresponding elements of the analogs.

As mentioned in Section 1 and in Section 2.3.1, most of the mapping models (e.g., [7], [22], [21], [24], [6], [16]) use at the final stage a localist mapping network (real or virtual) or its symbolic analog. However, in HRRs and BSCs, another technique is used to find corresponding elements of analogs ([39], [42], [27], [28]). This technique is based on operations that work directly on the distributed representations of analogs.

For example, in order to find the entity in various episodes corresponding to Jane in probe, the following procedure is used [42]: First, noisy roles of Jane are extracted from the probe by unbinding: $\mathbf{jane\text{-}roles} = \langle \mathbf{P} * \mathbf{jane}^{\mathrm{T}} \rangle + \mathrm{noise}$. Then, noisy fillers of these roles are extracted from the considered episode $\mathbf{E}$, again using unbinding: $\langle \mathbf{E} * \mathbf{jane\text{-}roles}^{\mathrm{T}} \rangle$. Finally, the similarity of those fillers with the episode entities is calculated by a dot product. For the LS episode, the entities are John and Fido, therefore the dot products $\langle \mathbf{E} * \mathbf{jane\text{-}roles}^{\mathrm{T}} \rangle \bullet \mathbf{john}$ and $\langle \mathbf{E} * \mathbf{jane\text{-}roles}^{\mathrm{T}} \rangle \bullet \mathbf{fido}$ are calculated. The entity corresponding to Jane should produce the largest dot product.

In [42], this procedure produced correct results for the LS and the AN episodes. For the SF, CM, and FOR episodes, the results were incorrect or ambiguous because, in those episodes, the similarity of role-filler bindings are inconsistent with the similarity of entities. More sophisticated versions of the mapping procedure are needed in order to resolve this ambiguity, e.g., "intermediate clean-up" [42].

5.2.1 Mapping by Similarity Estimation in APNNs

In order to map the elements of two analogs, first their elements of all levels of hierarchy (or chunks) should be encoded using the APNN representations. Here, let us use the role-filler representation scheme of (20), (21), and (22) (see also Table 1).

As proposed in Section 4.1, six hierarchical levels (base-level plus level #0 – level #4) can be distinguished in the episodes represented by the chosen scheme. The simplest version of mapping we discuss here is to try putting into correspondence the elements of episodes at all hierarchical levels by direct comparison of the codevectors of those elements.

Let us consider the similarity matrices calculated for the elements of the probe episode and one of the other episodes (Tables 4, 5, and 6). The matrix elements are the overlaps of codevectors of various chunks. It can be seen that for the LS and the AN episodes (Tables 4 and 5), the matrix elements of the main diagonal are the largest. It means that the corresponding elements of these episodes and the probe correctly map to each other. Actually, only the similarity between the codevectors inside each level should be taken into account, as indicated by the boxes in the tables. Finally, the entities of level #1 that are the components of the corresponding bindings of level #2 should be also marked as corresponding (irrespective of their similarity).

For the CM, the SF (Table 6), and the FOR episodes, the "correct" correspondence with the probe cannot be established by the similarity of chunks. For these episodes, the similarity of the agent/object roles for bite/flee relations either comes into contradiction with the similarity of those for the probe (the SF and the FOR episodes) or into contradiction with the similarity of their fillers (the CM episode). These results of finding correspondences agree with the results of Plate [42] obtained without "intermediate clean-up." It should be noted that such types of analogs are difficult to map even for people. To establish "correct" correspondences in the considered cases, more elaborated APNN representations or more "formal" techniques for finding correspondences should be used (see also discussion below).

TABLE 5
The Similarity Matrix for the Elements of the Probe and the AN Episode.

		level#4	level#3		Level#2		level#1			
		Probe	**Antc**	**Cnsq**	**Bite**	**Flee**	**bite_a**	**bite_o**	**flee_a**	**flee_o**
level#4	**Epis(AN)**	0.25	0.19	0.19	0.08	0.08	0.06	0.07	0.07	0.07
level#3	**Antc**	0.19	0.33	0.02	0.11	0.03	0.10	0.10	0.02	0.02
	Cnsq	0.20	0.02	0.35	0.02	0.12	0.02	0.03	0.11	0.11
level#2	**Bite**	0.07	0.11	0.03	0.16	0.03	0.15	0.14	0.02	0.02
	Flee	0.07	0.02	0.11	0.02	0.16	0.02	0.02	0.14	0.14
level#1	**bite_a**	0.06	0.10	0.02	0.15	0.02	0.26	0.02	0.02	0.01
	bite_o	0.06	0.09	0.02	0.13	0.02	0.02	0.25	0.02	0.02
	flee_a	0.06	0.02	0.10	0.02	0.14	0.02	0.02	0.26	0.02
	flee_o	0.07	0.02	0.09	0.02	0.13	0.01	0.03	0.02	0.25

Notations are the same as in Table 1.

TABLE 6
The Similarity Matrix for the Elements of the Probe and the SF Episode

		level#4	level#3		level#2		level#1			
		Probe	**Antc**	**Cnsq**	**Bite**	**Flee**	**bite_a**	**bite_o**	**flee_a**	**flee_o**
level#4	**Epis(SF)**	0.38	0.28	0.29	0.19	0.20	0.15	0.14	0.15	0.15
level#3	**Antc**	0.28	0.33	0.21	0.10	0.28	0.09	0.09	0.19	0.19
	Cnsq	0.27	0.21	0.34	0.28	0.13	0.20	0.20	0.11	0.09
level#2	**Flee**	0.19	0.12	0.28	0.16	0.42	0.13	0.14	0.28	0.29
	Bite	0.19	0.28	0.11	0.42	0.17	0.28	0.29	0.15	0.13
level#1	**flee_a**	0.14	0.11	0.20	0.14	0.29	0.02	0.25	0.56	0.02
	flee_o	0.15	0.09	0.20	0.13	0.29	0.25	0.02	0.02	0.56
	bite_a	0.14	0.20	0.09	0.28	0.13	0.56	0.02	0.02	0.25
	bite_o	0.14	0.20	0.09	0.28	0.13	0.02	0.56	0.26	0.02

Notations are the same as in Table 1.

6 Discussion

Estimation of the similarity of structures is a key procedure in models of cognition and in other systems operating with structured information. Realization of this procedure is determined by the employed scheme for structure representation.

In the past, mainly symbolic or localist connectionist representations were used for structure encoding. Those representations suffer from the scaling problem. Each combination of items requires the allocation of new memory for its representation as a local unit or a symbol string.

An even more pronounced problem is that localist and symbolic representations are semantically brittle or inflexible. For example, whenever a new information item is represented, it is necessary to decide if new units should be allocated to represent it and its constituents or whether similar existing representations can be exploited. Since localist representations are indivisible, it is difficult to represent a gradual degree of similarity or difference between them. A related problem for localist and symbolic representations is the estimation of structure similarity, which requires computationally very expensive procedures.

The problems of structure representation and processing with nondistributed representations are exemplified by the models of analogical reasoning. It should be admitted that in recent models of analogical mapping (LISA [24], STAR2 [16], and especially DRAMA [6]), attempts are made to use the semantic flexibility of distributed representations. However, at some stage of implementation, these models return to the versions of localist networks that are conceptually similar to the structure handling techniques of earlier prominent models (SME [7], ACME [53], Copycat [21]) based on nondistributed representations.

The computationally expensive stage of structure matching leads to the necessity of introducing a two-stage process for the retrieval of analogs from memory. Since a lot of potentially analogous situations are stored in memory, it becomes prohibitive to use an expensive process in comparing the probe with each of them. Therefore, a common strategy is to use a computationally cheap process (which does not take structure into account) for the selection of the candidates, and then to use a complex computation for estimation of the similarity between the probe and the candidates (taking into account all aspects of similarity), as in MAC/FAC [9] and ARCS [53].

In APNNs, HRRs, and BSCs, it has been possible to create an on-the-fly scheme for distributed structure representation, where both the set of components and their arrangements influence the similarity of codevectors. Therefore, similar structures produce similar codevectors. The similarity of codevectors is measured just by the overlap of their 1s (APNNs and BSCs) or dot product (HRRs). Thus, it is not necessary to determine the correspondence between the elements of two structures in order to estimate their overall similarity.

A single-stage process of estimating similarity of structures by the overlap of their APNN-style codevectors was tested on a simple analogical retrieval task. For both schemes of structure representation that were applied, the order of scores for the episodes with different types of similarity corresponded to the order of HRR scores for the same task [42] and to the retrieval pattern observed in people. A connection between the codevector similarity and the probability of human analogical access can be seen as follows: Increased similarity between representations of the episode and the probe increases the probability of the episode recall from long-term memory. If long-term memory is implemented as an associative memory and probe codevectors are somewhat distorted versions of stored ones, this could account for such a behavior and also for some empirical phenomena concerning analogical access that were summarized in Table 1 of [24]. In particular, it concerns the phenomena of competitive access and easier access of close analogs. Easier access to more abstract schemas can also be explained by the property of the CDT procedure to produce more similar codevectors at lower thinning depth. The phenomenon that highly familiar analogs are more likely to be retrieved even if they are less similar to the probe than alternatives can be explained by their larger memory traces (produced, e.g., by the Hebbian learning rule) that facilitate retrieval.

The range of possible schemes for structure representations in APNNs is extended by the employed binding procedure and various schemes of order representation. For example, both a role-filler scheme and a predicate-arguments scheme can be used for representation of relational structure.

Each chunk (a composite item, a substructure, a structure) is represented in the APNN scheme by a thinned codevector in which the component codevectors of the lower hierarchical level are bound together. This allows one to build the codevector of a higher-level item simply by superimposing and binding together (by thinning) the component codevectors. Therefore, it is not necessary to exclusively use the role-filler scheme where the roles of the higher level should be determined in order to make bindings with the chunks of the previous hierarchical level. It also decreases the risk of "spurious memories" when an outer-product superposition learning rule is used to store the codevectors of chunks.

In APNNs, the compositional (part-whole) hierarchy of representations receives primary attention. The chunk or proposition of each complexity level is stored and processed in a separate memory array. To retrieve the components of a composite codevector, the similarity with the codevectors of the lower hierarchical level is used. Also, this hierarchy simplifies the mapping of the elements of analogs. The correspondence should be primarily searched between the elements of the same hierarchical level.

The simplest mapping technique is to put into correspondence the elements of two analogs of the same hierarchical level which have the largest overlap of codevectors. For the encoding of entities and propositions chosen in this paper to represent animal stories, this technique worked successfully for the episodes with literal and analogical similarity. For the interpretation of more formal analogies, other representations could be tried; for example, another feature structure of the base-level components or different representation scheme for propositional structure. However, the present results can already account for the phenomena from Table 1 of [24] that semantic similarity has a greater impact on access than on mapping (and the reverse for isomorphism). The CM and SF episodes are semantically more similar to the probe than the AN episode and produced higher similarity scores to the probe thus suggesting an easier access. However, during mapping, the analogical elements at all hierarchical levels of the isomorphic AN episode were correctly mapped to those of the probe, whereas this was not observed for the nonisomorphic CM and SF episodes.

A more structure-sensitive technique for finding correspondences between the elements of analogs is to decode the chunks (starting from the top level) through their subchunks of the lower level, to find the correspondences between the roles, and to put into correspondence their fillers—irrespective of their similarity.

Another possible technique of mapping is the substitution of identical codevectors for the corresponding elements of analogs and reencoding the analogs. This allows for use of the information about established correspondences in order to find other correspondences. Also, it can be used to verify the consistency of mapping: If the elements of all levels are mapped correctly and the structure of propositions is identical, then identical codes will result for the whole structures.

Such techniques will be considered in greater detail elsewhere. They include essentially sequential steps that provide flexibility to the mapping process—though they may alternate with parallel stages, e.g., finding the most similar codevectors. The process of mapping can be substantially sequential because it probably works only on a single pair of analogs simultaneously. In contrast, the process of analogical access should allow essentially parallel implementation because it requires fast retrieval from a potentially very huge base of analogical episodes.

In this paper, as in [42], only the simplest flat feature structures (of two features) were used for entities and for relational agents/objects (of a single feature). Such descriptions were generated artificially. Actually, the items of the lowest level may have a complex structure formed in the interaction with environment. For example, the features of *type* are categories that themselves possess a complex hierarchical structure [35]. Therefore, the transition to more realistic problems will require dealing with the problems concerning the representation of meaning and symbol grounding ([24], [29], [42], see also [5], [2], [18], [4]). Related potential problems of how similar items from diverse conceptual and linguistic structures might be uniformly allocated to the proper levels of part-whole hierarchy and how the items that seem cross levels might be handled also need further investigation.

Further work should reveal how potentially attractive, from the scaling point of view, schemes for distributed structure representation, processing algorithms exploiting them, and implementations of long-term memory will actually behave themselves for very large numbers of structured items. The APNN scheme seems more neurologically relevant compared to other schemes since it uses distributed sparse binary codes. Besides, manipulations with such codes are very simple—and memory for their storage becomes cheaper.

7 Conclusion

The representation of structures with sparse binary codevectors in the framework of hierarchical architecture of Associative-Projective Neural Networks permits the estimation of their aggregate similarity by the overlap of corresponding codevectors. Such a representation is not neurologically implausible and permits simple and fast implementation.

The main procedure for codevector construction is that of Context-Dependent Thinning. This procedure binds the codevectors of items together and maintains a low density of codevectors of various complexity. This allows distributed auto-associative memories with a superposition learning rule to be considered as a storage medium for the codevectors of substructures of various hierarchical levels.

This "thinning" procedure, as well as some others, is also used to represent the order of arguments in a relation. This offers a potential for the encoding of various types of data structures, including role-filler and predicate-arguments schemes for representation of relational structures.

The similarity of codevectors representing analogous episodes was applied to the modeling of analogical retrieval from memory. The similarity of codevectors can be also used for analogical mapping, however, "difficult" cases may require more complex techniques, including similarity-based expansion of higher-level codevectors through their component codevectors.

The concepts and techniques described in this paper demonstrate the potential of distributed representations for the problem of analogy as well as for other structure-sensitive problems of AI that were assumed to be solvable only with symbolic or localist representations.

Acknowledgments

The author would like to thank Ernst Kussul for insightful discussions as well as Tony Plate, Pentti Kanerva, and the anonymous reviewers for their helpful and extensive comments and suggestions. This work was supported in part by the International Science Foundation Grants U4M000 and U4M200.

References

[1] S. Amari, "Characteristics of Sparsely Encoded Associative Memory," *Neural Networks,* vol. 2, pp. 445-457, 1989.

[2] L.W. Barsalou, "Perceptual Symbol Systems," *Behavioral and Brain Sciences,* vol. 22, pp. 577-609, 1999.

[3] E.B. Baum, J. Moody, and F. Wilczek, "Internal Representations for Associative Memory," *Biological Cybernetics 59,* pp. 217-228, 1988.

[4] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R.A. Harshman, "Indexing by Latent Semantic Analysis," *J. Am. Soc. for Information Science,* vol. 41, no. 6, pp. 391-407, 1990.

[5] G. Dorffner and E. Prem, "Connectionism, Symbol Grounding, and Autonomous Agents," Technical Report TR-93-17, Austrian Research Inst. for AI, 1993.

[6] C. Eliasmith and P. Thagard, "Integrating Structure and Meaning: A Distributed Model of Analogical Mapping," *Cognitive Science,* vol. 25, no. 1, 2001.

[7] B. Falkenhainer, K.D. Forbus, and D. Gentner, "The Structure-Mapping Engine: Algorithm and Examples," *Artificial Intelligence,* vol. 41, pp. 1-63, 1989.

[8] J.A. Feldman, "Neural Representation of Conceptual Knowledge," *Neural Connections, Mental Computation,* L. Nadel, L.A. Cooper, P. Culicover, and R.M. Harnish, eds., pp. 68-103, 1989.

[9] K.D. Forbus, D. Gentner, and K. Law, "MAC/FAC: A Model of Similarity-Based Retrieval," *Cognitive Science,* vol. 19, no. 2, pp. 141-205, 1995.

[10] P. Frasconi, M. Gori, and A. Sperduti, "A General Framework for Adaptive Processing of Data Structures," Technical Report DSI-RT-15/97, Universita degli Studi di Firenze, Dipartimento di Sistemi e Informatica, Firenze, Italy, 1997.

[11] A.A. Frolov, D. Husek, and I.P. Muraviev, "Information Capacity and Recall Quality in Sparsely Encoded Hopfield-like Neural Network: Analytical Approaches and Computer Simulation," *Neural Networks,* vol. 10, pp. 845-855, 1997.

[12] R.W. Gayler, "Multiplicative Binding, Representation Operators, and Analogy," *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences,* K. Holyoak, D. Gentner, and B. Kokinov, eds., Sofia, Bulgaria: New Bulgarian University, p. 405, 1998 (poster abstract; full poster available at: http://cogprints.soton.ac.uk/abs/comp/199807020).

[13] D. Gentner, "Structure-Mapping: A Theoretical Framework for Analogy," *Cognitive Science,* vol. 7, pp 155-170, 1983.

[14] D. Gentner and A.B. Markman, "Analogy-Based Reasoning," *Handbook of Brain Theory and Neural Networks,* M.A. Arbib, ed., pp. 91-93, Cambridge, Mass.: MIT Press, 1995.

[15] D. Gentner and A.B. Markman, "Structure Mapping in Analogy and Similarity," *Am. Psychologist,* vol. 52, no. 1, pp. 45-56, 1997.

[16] B. Gray, G.S. Halford, W.H. Wilson, and S. Phillips, "A Neural Net Model for Mapping Hierarchically Structured Analogs," *Proc. Fourth Conf. Australasian Cognitive Science Soc.,* Sept. 1997.

[17] G.S. Halford, W.H. Wilson, and S. Phillips, "Processing Capacity Defined by Relational Complexity: Implications for Comparative, Developmental, and Cognitive Psychology," *Behavioral and Brain Sciences,* vol. 21, pp. 723-802, 1998.

[18] S. Harnad, "The Symbol Grounding Problem," *Physica D,* vol. 42, pp. 335-346, 1990.

[19] D.O. Hebb, *The Organization of Behavior.* New York: Wiley, 1949.

[20] G.E. Hinton, "Mapping Part-Whole Hierarchies into Connectionist Networks," *Artificial Intelligence,* vol. 46, pp. 47-76, 1990.

[21] D.R. Hofstadter and M. Mitchell, "Conceptual Slippage and Mapping: A Report of the Copycat Project," *Proc. 10th Ann. Conf. Cognitive Science Soc.,* pp. 601-607, 1988.

[22] K.J. Holyoak and P. Thagard, "Analogical Mapping by Constraint Satisfaction," *Cognitive Science,* vol. 13, pp. 295-355, 1989.

[23] J.J. Hopfield, D.I. Feinstein, and R.G. Palmer, "Unlearning has a Stabilizing Effect in Collective Memories," *Nature,* vol. 304, pp. 158-159, 1983.

[24] J.E. Hummel and K.J. Holyoak, "Distributed Representations of Structure: A Theory of Analogical Access and Mapping," *Psychological Rev.,* vol. 104, pp. 427-466, 1997.

[25] P. Kanerva, *Sparse Distributed Memory.* MIT Press, 1988.

[26] P. Kanerva, "Binary Spatter-Coding of Ordered K-Tuples," *Proc. Int'l Conf. Artificial Neural Networks—ICANN '96,* C. von der Malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff, eds., pp. 869-873, 1996.

[27] P. Kanerva, "Fully Distributed Representation," *Proc. 1997 Real World Computing Symp. (RWC '97),* pp. 358-365, 1997.

[28] P. Kanerva, "Dual Role of Analogy in the Design of a Cognitive Computer," *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences (workshop proc. Analogy '98),* K. Holyoak, D. Gentner, and B. Kokinov, eds., pp. 164-170, 1998.

[29] E.M. Kussul, *Associative Neuron-Like Structures.* Kiev: Naukova Dumka, 1992 (in Russian).

[30] E.M. Kussul and T.N. Baidyk, "Design of a Neural-Like Network Architecture for Recognition of Object Shapes in Images," *Soviet J. Automation and Information Sciences,* vol. 23, no. 5, pp. 53-58, 1990.

[31] E.M. Kussul and T.N. Baidyk, "On Information Encoding in Associative-Projective Neural Networks," (Preprint 93-3). Kiev, Ukraine: V.M. Glushkov Inst. of Cybernetics, 1993 (in Russian).

[32] E.M. Kussul and D.A. Rachkovskij, "Multilevel Assembly Neural Architecture and Processing of Sequences," *Neurocomputers and Attention: Vol. II Connectionism and Neurocomputers,* A.V. Holden and V.I. Kryukov, eds., pp. 577- 590, 1991.

[33] E.M. Kussul, D.A. Rachkovskij, and T.N. Baidyk, "Associative-Projective Neural Networks: Architecture, Implementation, Applications," *Proc. Fourth Int'l Conf. Neural Networks and Their Applications,* pp. 463-476, Nov. 1991.

[34] A. Lansner and O. Ekeberg, "Reliability and Speed of Recall in an Associative Network," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 7, pp. 490-498, 1985.

[35] A.B. Markman, "Structural Alignment in Similarity and Its Influence on Category Structure," *Cognitive Studies,* vol. 4, no. 4, pp. 19-37, 1997.

[36] P.M. Milner, "A Model for Visual Shape Recognition," *Psychological Rev.,* vol. 81, pp. 521-535, 1974.

[37] G. Palm, "On Associative Memory," *Biological Cybernetics,* vol. 36, pp. 19-31, 1980.

[38] G. Palm and T. Bonhoeffer, "Parallel Processing for Associative and Neuronal Networks," *Biological Cybernetics,* vol. 51, pp. 201-204, 1984.

[39] T.A. Plate, "Estimating Structural Similarity by Vector Dot Products of Holographic Reduced Representations," *Advances in Neural Information Processing Systems 6 (NIPS '93),* J.D. Cowan, G. Tesauro, and J. Alspector, eds., pp. 1109-1116, 1994.

[40] T.A. Plate, "Holographic Reduced Representations," *IEEE Trans. Neural Networks,* vol. 6, pp. 623-641, 1995.

[41] T. Plate, "A Common Framework for Distributed Representation Schemes for Compositional Structure," *Connectionist Systems for Knowledge Representation and Deduction,* F. Maire, R. Hayward, and J. Diederich, eds., pp. 15-34, 1997.

[42] T.A. Plate, "Analogy Retrieval and Processing with Distributed Vector Representations," *Expert Systems: The Int'l J. Knowledge Eng. and Neural Networks,* special issue on Connectionist Symbol Processing, vol. 17, no. 1, pp. 29-40, 2000.

[43] J.B. Pollack, "Recursive Distributed Representations," *Artificial Intelligence,* vol. 46, pp. 77-105, 1990.

[44] D.A. Rachkovskij, "On Numerical-Analytical Investigation of Neural Network Characteristics," *Neuron-Like Networks and Neurocomputers,* pp. 13-23, Kiev, Ukraine: V.M. Glushkov Inst. of Cybernetics, 1990 (in Russian).

[45] D.A. Rachkovskij, "Development and Investigation of Multilevel Assembly Neural Networks," Unpublished PhD dissertation, Kiev, Ukrainian SSR: V.M. Glushkov Inst. of Cybernetics, 1990 (in Russian).

[46] D.A. Rachkovskij and E.M. Kussul, "Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning," *Neural Computation,* vol. 13, no. 2, pp. 411-452, 2001 (paper draft available at http://cogprints.soton.ac.uk/, ID code: cog00001240).

[47] B. Ross, "Distinguishing Types of Superficial Similarities: Different Effects on the Access and Use of Earlier Problems," *J. Experimental Psychology: Learning, Memory, and Cognition,* vol. 15, pp. 456-468, 1989.

[48] L. Shastri and V. Ajjanagadde, "From Simple Associations to Systematic Reasoning: Connectionist Representation of Rules, Variables, and Dynamic Bindings using Temporal Synchrony," *Behavioral and Brain Sciences,* vol. 16, pp. 417-494, 1993.

[49] G. Sjodin, "The Sparchunk Code: A Method to Build Higher-Level Structures in a Sparsely Encoded SDM," *Proc. Int'l Joint Conf. Neural Networks '98,* pp. 1410-1415, 1998.

[50] G. Sjodin, P. Kanerva, B. Levin, and J. Kristoferson, "Holistic Higher-Level Structure-Forming Algorithms," *Proc. 1998 Real World Computing Symp.—RWC '98,* pp. 299-304, 1998.

[51] P. Smolensky, "Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems," *Artificial Intelligence,* vol. 46, pp. 159-216, 1990.

[52] A. Sperduti, "Labeling RAAM," *Connection Science,* vol. 6, pp. 429-459, 1994.

[53] P. Thagard, K.J. Holyoak, G. Nelson, and D. Gochfeld, "Analog Retrieval by Constraint Satisfaction," *Artificial Intelligence,* vol. 46, pp. 259-310, 1990.

[54] M.V. Tsodyks, "Associative Memory in Neural Networks with the Hebbian Learning Rule," *Modern Physics Letters B,* vol. 3, pp. 555-560, 1989.

[55] D.S. Touretzky, "BoltzCONS: Dynamic Symbol Structures in a Connectionist Network," *Artificial Intelligence,* vol. 46, pp. 5-46, 1990.

[56] A.A. Vedenov, "Spurious Memory," *Model Neural Networks,* Moscow: I.V. Kurchatov Inst. of Atomic Energy (preprint IAE-4395/1), 1987.

[57] C. von der Malsburg, "Am I Thinking Assemblies?" *Proc. 1984 Trieste Meeting on Brain Theory,* G. Palm and A. Aertsen, eds., pp. 161-176, 1986.

[58] C.M. Wharton, K.J. Holyoak, P.E. Downing, T.E. Lange, T.D. Wickens, and E.R. Melz, "Below the Surface: Analogical Similarity and Retrieval Competition in Reminding," *Cognitive Psychology,* vol. 26, pp. 64-101, 1994.

[59] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins, "Non-Holographic Associative Memory," *Nature,* vol. 222, pp. 960-962, 1969.

Dmitri A. Rachkovskij received the MS degree in radiophysics and electronics from Rostov State University, Rostov-on-Don, Russia, in 1983, and the PhD degree from the V.M. Glushkov Cybernetics Center, Kiev, Ukraine, in 1990. From 1983 to 1987, he worked as a R&D engineer in computer-based systems for Rostov Special R&D Bureau. In 1987, he became a PhD student at the Cybernetics Center under the supervision of Academician N.M. Amosov, who since the early 60s has led one of the most interesting research programs in AI, and E.M. Kussul, who made a major contribution to that program by inventing the paradigm of Associative-Projective Neural Networks (APNNs). Currently, Dr. Rachkovskij is a senior researcher at the Cybernetics Center. His research has been connected with APNNs, in particular, with the schemes for binary sparse distributed encoding of heterogeneous information, the properties and information capacity of distributed auto-associative memory with the low activity level, the mechanisms for the representation and processing of complex sequences in hierarchical APNNs. He also worked extensively in the areas of classification, invariant pattern recognition, and micromechanics. He developed algorithms for a number of applications, including acoustical and image recognition (texture, handprinted character, OCR, handwritten text), as well as information retrieval and filtering. He also took part in the development of software and hardware for associative-projective neurocomputers. Currently, his work is focused on the representation and processing of hierarchical structures with APNNs.