Experiments on Preserving Pieces of Information in a Given Order in Holographic Reduced Representations and the Continuous Geometric Algebra Model

Agnieszka Patyk-Łońska
Gdańsk University of Technology
ul. Narutowicza 11/12, Gdańsk 80-233, Poland
patyk@pg.gda.pl, http://www.pg.gda.pl/ patyk

Keywords: distributed representations of data, geometric algebra, HRR, BSC, word order, trajectory associations, bag of words

Received: October 23, 2011

Geometric Analogues of Holographic Reduced Representations (GA_c, which is the continuous version of the previously developed discrete GA model) employ role-filler binding based on geometric products. Atomic objects are real-valued vectors in n-dimensional Euclidean space and complex statements belong to a hierarchy of multivectors. The property of GA_c and HRR studied here is the ability to store pieces of information in a given order by means of trajectory association. We describe results of three experiments: finding correct item or correct place of an item in a sequence and finding the alignment of items in a sequence without the precise knowledge of trajectory vectors.

Povzetek: Članek preučuje ohranitev informacij pri obliki holografske hrambe podatkov.

1 Introduction

The work presented here is a result of experimenting with trajectory association technique using the newly-developed distributed representation model GA_c [20].

An ideal distributed representation system needs to meet several criteria in order to successfully perform cognitive tasks. These include computational efficiency, noise tolerance, scaling and ability to represent complex structures. The most widely used definition of a distributed representation of data is due to Hinton *et al.* [13]: in a *distributed representation of data* each concept is represented over a number of units and each unit participates in the representation of some number of concepts. The size of a distributed representation is usually fixed and the units have either binary or continuous-space values. In most distributed representations only the overall pattern of activated units has a meaning.

Such patterns of activity are hard to understand and interpret, therefore they are often compared to greyscale images. Distributed representations usually take the form of one-dimensional vectors, while greyscale images are two-dimensional matrices, but the way the pixels are aligned (one-dimensional string or two-dimensional array) is of no relevance. Since the information is distributed over the elements of a vector, a great percentage of units ("pixels") can be changed without making the vector (overall "picture") unrecognizable.

Let us consider an example of storing the following information: "Fido bit Pat". The action in this statement is $bite$ and the features (i.e. $roles$) of this action are an agent and an object, denoted $bite_{agt}$ and $bite_{obj}$, while their $fillers$ are $Fido$ and Pat respectively. If we consider storing the way that the action is performed, we can add a third feature ($role$), e.g. $bite_{way}$. If we store $Fido$, Pat, $bite_{agt}$ and $bite_{obj}$ as vectors, we are able to encode "Fido bit Pat" as

$$bite_{agt} * Fido + bite_{obj} * Pat.$$

The operation of $binding$, denoted by "$*$", takes two vectors and produces another vector, often called a $chunk$ of a sentence. It would be ideal for the resulting vector not to be similar to the original vectors but to have the same dimensions as the original vectors. $Superposition$, denoted by "$+$", is an operation that takes any number of vectors and creates another one that is similar to the original vectors. Usually, the superimposed vectors are already the result of the binding operation.

For more details and examples on distributed representations of data the reader should refer to [20].

2 Preserving Pieces of Information in a Given Order

While some solutions to the problem of preserving pieces of information in a given order have proved ingenious, others are obviously flawed. Let us consider the representation of the word eye — it has three letters, one of which occurs

This paper is based on A. Patyk-Łońska *Preserivng pieces of information in a given order in HRR and GA_c* published in the proceedings of the 1st International Workshop on Advances in Semantic Information Retrieval (part of the FedCSIS'2011 conference).

twice. The worst possible choice of binding and superposition would be to store quantities of letters, e.g.

$$eye = twice * e + once * y,$$

since we would not be able to distinguish eye from eey or yee. Another ambiguous representation would be to remember the neighborhood of each letter

$$eye = before_y * e + between_e * y + after_y * e.$$

Unfortunately, such a method of encoding causes words eye and $eyeye$ to have the same representation

$$\begin{aligned} eyeye &= before_y * e + 2 \cdot between_e * y + \\ &\quad (before_y + after_y) * e + after_y * e \\ &= 2\big(before_y * e + between_e * y \\ &\quad + after_y * e\big) \\ &= 2\, eye. \end{aligned}$$

Real-valued vectors are normalized in most distributed representation models, therefore the factor of 2 would be most likely lost in translation. Such $contextual\ roles$ (Smolensky [25]) cause problems when dealing with certain types of palindromes. Remembering positions of letters is also not a good solution

$$eye = letter_{first} * e + letter_{second} * y + letter_{third} * e$$

as we need to redundantly repeat the first letter as the third letter, otherwise we could not distinguish eye from ey or ye. Secondly, this method of encoding will not detect similarity between eye and $yeye$.

Pike argues in [21] that matrix-based memory is multidirectional, i.e. it allows both forward and backward association — having two vectors a and b and their binding $M = ab$ we can extract both a and b by performing a reverse operation on the appropriate side of the matrix. Convolution-correlation systems, on the other hand, regard bindings $a \circledast b$ and $b \circledast a$ as identical. We will use a similar technique, asking right-hand-side and left-hand-side questions during experiments described in the following sections.

A quantum-like attempt to tackle the problem of information ordering was made in [1] — a version of semantic analysis, reformulated in terms of a Hilbert-space problem, is compared with structures known from quantum mechanics. In particular, an LSA matrix representation ([1, 10]) is rewritten by the means of quantum notation. Geometric algebra has also been used extensively in quantum mechanics ([2, 4, 3]) and so there seems to be a natural connection between LSA and GA_c, which is the ground for fututre work on the problem of preserving pieces of information in a given order.

As far as convolutions are concerned, the most interesting approach to remembering information in a given order has been described in [12]. Authors present a model that builds a holographic lexicon representing both word meaning and word order from unsupervised experience with natural language texts comprising altogether 90000 words. This model uses simple convolution and superposition to construct n-grams recording the frequency of occurrence of every possible word sequence that is encountered, a window of about seven words around the target word is usually taken into consideration. To predict a word in a completely new sentence, the model looks up the frequency with which the potential target is surrounded by words present in the new sentence. To be useful, n-gram models need to be trained on massive amounts of text and therefore require extensive storage space. We will use a completely different approach to remembering information order — trajectory association described by Plate in [23]. Originally, this technique also used convolution and correlation, but this time items stored in a sequence are actually superimposed, rather than being bound together.

3 Trajectory Associacion

In the HRR model vectors are normalized and therefore can be regarded as radii of a sphere of radius 1. If we attach a sequence of items, say A, B, C, D, E to arrowheads of five of those vectors, we obtain a certain $trajectory$ associated with sequence $ABCDE$. This is a geometric analogue to the $method\ of\ loci$ which instructs to remember a list of items by associating each term with a distinctive location along a familiar path. Let k be a randomly chosen HRR vector and let

$$k^i = k \circledast k^{i-1} = k^{i-1} \circledast k, \quad i > 1$$

be its ith $power$, with $k^1 = k$. The sequence S_{ABCDE} is then stored as

$$S_{ABCDE} = A \circledast k + B \circledast k^2 + C \circledast k^3 + D \circledast k^4 + E \circledast k^5.$$

Of course, each power of k needs to be normalized before being bound with a sequence item. Otherwise, every subsequent power of k would be larger or smaller than its predecessor. As a result, every subsequent item stored in a sequence would have a bigger or a smaller share in vector S_{ABCDE}. Obviously, this method cannot be applied to the discrete GA model or to BSC, since it is impossible to obtain more than two distinct powers of a vector with the use of XOR as a means of binding.

This technique has a few obvious advantages present in HRR but not in GA_c had we wished to use ordinary vectors as first powers — different powers of a vector k would then be multivectors of different ranks. While k^i and $k^{i\pm1}$ are very similar in HRR, in GA_c they would not even share the same blades. Further, the similarity of k^i and k^{i+m} in HRR is the same as the similarity of k^j and k^{j+m}, whereas in GA_c that similarity would depend on the parity of i and j. In the light of these shortcomings, we need to use another structure acting as a first power in order to make trajectories work in GA_c. Let t be a random normalized full

multivector over $\mathbb{R}^n$ and let us define powers of t in the following way

$$\begin{aligned} t^1 &= t, \\ t^i &= (t^{i-1})t \text{ for } i > 1. \end{aligned}$$

We will store vectors $a_1 \dots a_l$ in a sequence $S_{a_1 \dots a_l}$ using powers of the multivector t

$$S_{a_1 \dots a_l} = a_1 t + a_2 t^2 + \dots + a_l t^l.$$

To answer a question "*What is the second item in a sequence?*" in GA_c we need to use the projected product

$$\langle S_{a_1 \dots a_l} (t^2)^+ \rangle_1 \approx a_2,$$

and to find out the place of item a_i we need to compute

$$(a_i)^+ S_{a_1 \dots a_l} \approx t^i.$$

Some may argue that such encoding puts a demand on items in the clean-up memory to hold information if they are roles or fillers, which is dangerously close to employing fixed data slots present in localist architectures. Actually, elements of a sequence can be recognized by their size, relatively shorter than the size of multivector t and its powers.

We present three experiments using trajectory association and we comment on test results for HRR and GA_c models. Firstly, we studied if an item can be retrieved given a sequence and an appropriate power of t, and vice versa — if a sequence and an item can lead to the power of t associated with that item. Finally, we tested whether both HRR and GA_c models can find the alignment of items in a sequence without the precise knowledge of vector t or its powers. Since the normalization using the square root of the number of chunks proved very noisy in statements containing powers of the trajectory vector, we decided to improve the HRR model. The HRR vectors in our tests were normalized by dividing them with their magnitude.

4 Correct Place Detection

In this experiment we investigated if powers of a (multi)vector t carry enough information about the original t. During 1000 tests for (multi)vector sizes ranging from 2^4 to 2^8 we asked the following question for every sequence S (a permutation of letters $\{A, B, C, D, E\}$): "*Where is A?*"

$$S \sharp A = \begin{cases} S \circledast A^* \approx t^x & \text{in HRR}, \\ A^+ S \approx t^x & \text{in } GA_c. \end{cases}$$

This amounted to 120000 questions altogether. For the purpose of the experiment described in Section 6 we used the clean-up memory consisting of powers of t only.

Ideally, every position of the letter A should come up as the correct answer the same number of times. However, since high powers of t acquire noise, lower powers should be recognized more often as the correct answer. Indeed, Figure 1 shows that in HRR the frequencies of the powers of t align with t being recognized most often and t^5 being recognized least often. In GA_c the percentage diagrams for various powers of t lay close to each other and often intertwine, still the relationship between the powers of t is similar to the one observed in HRR.

Since lower powers of t are recognized correctly more often, higher powers of t come up more often as the incorrect answer to $S \sharp A$. Vector t^3 is the correct answer to $S_{\bullet\bullet A \bullet\bullet} \sharp A$. However, if t^3 is not recognized, the next most similar answer will be t^5 because it contains three "copies" of t^3, indicated here by brackets

$$\Big\{ t * \Big(t * [t \Big\} * t \Big) * t].$$

The second most similar item will be t^4 because it contains two "copies" of t, and so on. The item least similar to t^3 will be t. This relationship should be best observable in HRR, since the powers can be multiplied from either side. In GA_c the powers of t can be increased from one side only and the relation between them should be less visible. Figure 2 shows that high powers of t are recognized more often in cases when the proper answer is not recognized — we will use this relationship in an experiment described in Section 6.

5 Correct Item Detection

Here we tested if trajectory association allows us to ask "*What is the xth item in a sequence?*"

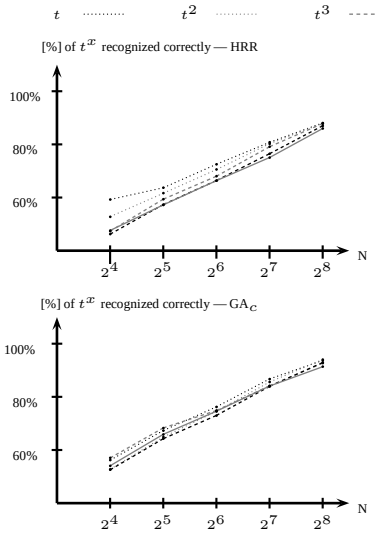
$$L_x \approx S \sharp t^x = \begin{cases} S \circledast (t^x)^* & \text{in HRR}, \\ S(t^x)^+ & \text{in } GA_c, \end{cases}$$

where $L_x \in \{A, B, C, D, E\}$ denotes the xth letter in a sequence. During 1000 tests for (multi)vector sizes ranging from 2^5 to 2^9 we asked that question for every permutation sequence of the set $\{A, B, C, D, E\}$, there were 120000 questions altogether for every (multi)vector t^x.

Again, we tested both HRR and GA_c models using a clean-up memory consisting only of expected answers, i.e. letters $\{A, B, C, D, E\}$. The results for both models (Figure 3) were similar with GA_c performing slightly better than HRR. In both models the first few letters of a sequence were more often recognized correctly than the last letters. Among the erroneously recognized letters, the last few letters of a sequence were most often offered as the most probable answer, which will come in handy in the next Section. The diagrams for GA_c lie closer together, once again indicating that trajectory association spreads information more evenly in GA_c than in HRR.

6 Item Alignment

The three previous tests were not very demanding for trajectory associations. Finally, we tested whether the HRR



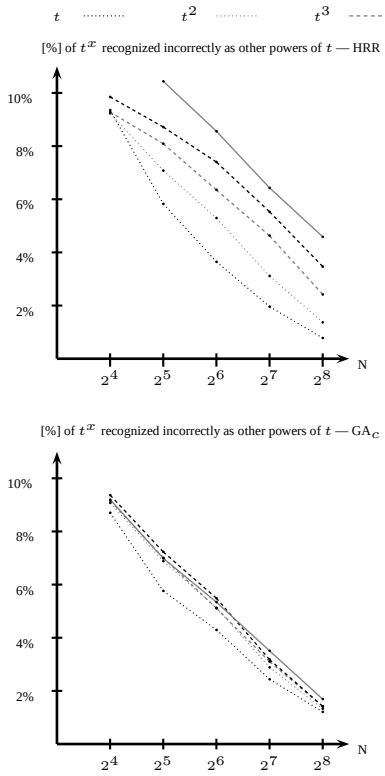
100% = $\frac{120000}{5}$ = 24000

	N=2^4	N=2^5	N=2^6	N=2^7	N=2^8
t	59.25%	63.67%	72.51%	80.77%	88.08%
t^2	52.80%	61.60%	70.52%	80.26%	87.93%
t^3	47.37%	59.37%	68.00%	79.11%	87.85%
t^4	46.22%	57.35%	66.40%	76.48%	87.01%
t^5	47.60%	57.23%	66.35%	75.05%	85.97%

100% = $\frac{120000}{5}$ = 24000

	$\mathbb{R}^4$	$\mathbb{R}^5$	$\mathbb{R}^6$	$\mathbb{R}^7$	$\mathbb{R}^8$
t	56.17%	67.31%	76.24%	86.73%	93.70%
t^2	52.72%	64.88%	74.53%	85.58%	94.05%
t^3	57.00%	68.30%	74.85%	84.13%	92.68%
t^4	52.53%	64.25%	73.00%	83.93%	92.90%
t^5	54.05%	65.89%	74.59%	84.06%	91.33%

Figure 1: Correct recognition of $S \sharp A \approx t^x$ in HRR and GA$_c$ using clean-up memory of $\{t, t^2, t^3, t^4, t^5\}$, 1000 trials.



100% = $\frac{120000}{5}$ = 24000

	N=2^4	N=2^5	N=2^6	N=2^7	N=2^8
t	9.35%	5.83%	3.65%	1.96%	0.78%
t^2	9.23%	7.08%	5.29%	3.11%	1.37%
t^3	9.28%	8.09%	6.35%	4.63%	2.42%
t^4	9.85%	8.72%	7.40%	5.53%	3.47%
t^5	11.63%	10.44%	8.56%	6.43%	4.59%

100% = $\frac{120000}{5}$ = 24000

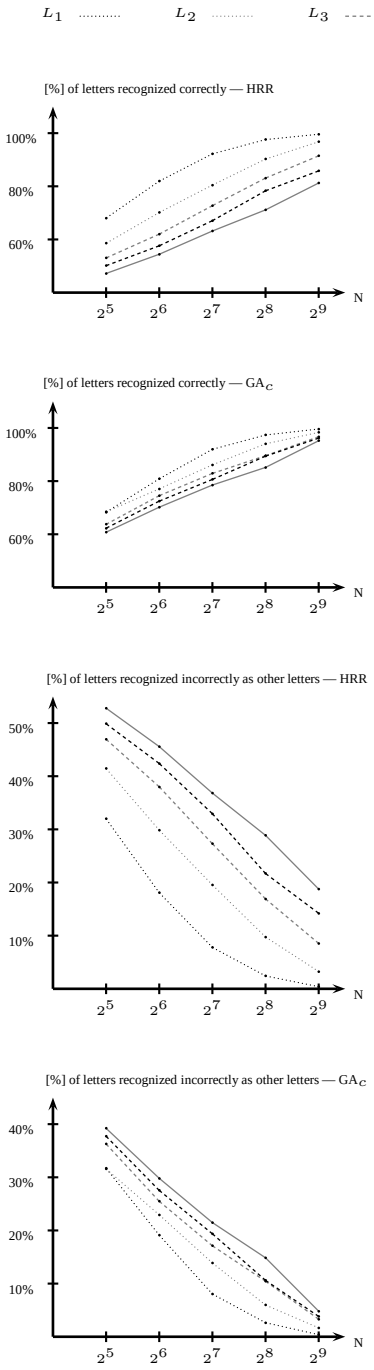
	$\mathbb{R}^4$	$\mathbb{R}^5$	$\mathbb{R}^6$	$\mathbb{R}^7$	$\mathbb{R}^8$
t	8.70%	5.76%	4.30%	2.44%	1.21%
t^2	9.08%	6.88%	5.13%	2.89%	1.33%
t^3	9.16%	7.01%	5.11%	3.10%	1.42%
t^4	9.37%	7.23%	5.49%	3.18%	1.42%
t^5	9.20%	6.99%	5.34%	3.51%	1.69%

Figure 2: Incorrect recognition of $S \sharp A \approx t^x$ in HRR and GA$_c$ using clean-up memory of $\{t, t^2, t^3, t^4, t^5\}$, 1000 trials.

and GA$_c$ models were capable of performing the following task:

> *Given only a set of letters A, B, C, D, E and an encoded sequence $S_{\bullet\bullet\bullet\bullet\bullet}$ comprised of those five letters find out the position of each letter in that sequence.*

We assumed that no direct access to t or its powers is given — they do belong to the clean-up memory, but cannot be retrieved “by name". One may think of this problem as a “black box" that inputs randomly chosen letter vectors and in return outputs a (multi)vector representing always the same sequence, irrespectively of the dimension of data. Inside, the black box generates (multi)vectors t, t^2, t^3, t^4, t^5. Their values are known to the observer but their names are not. Since we can distinguish letters from non-letters, the naive approach would be to try out all 120 alignments of letters A, B, C, D and E using all possible combinations of non-letters as the powers of t. Unfortunately, powers of



	N=2^5	N=2^6	N=2^7	N=2^8	N=2^9
L_1	68.03%	81.94%	92.21%	97.58%	99.59%
L_2	58.56%	70.15%	80.46%	90.25%	96.75%
L_3	53.05%	62.00%	72.71%	83.09%	91.46%
L_4	50.13%	57.61%	67.06%	78.31%	85.79%
L_5	47.20%	54.43%	63.18%	71.13%	81.24%

	$\mathbb{R}^5$	$\mathbb{R}^6$	$\mathbb{R}^7$	$\mathbb{R}^8$	$\mathbb{R}^9$
L_1	68.30%	80.88%	91.96%	97.36%	99.61%
L_2	68.43%	77.08%	86.12%	94.03%	98.33%
L_3	63.73%	74.52%	82.89%	89.56%	96.70%
L_4	62.28%	72.47%	80.63%	89.39%	96.18%
L_5	60.78%	70.19%	78.52%	85.15%	95.19%

	N=2^5	N=2^6	N=2^7	N=2^8	N=2^9
L_1	31.98%	18.07%	7.79%	2.42%	0.41%
L_2	41.44%	29.85%	19.54%	9.75%	3.25%
L_3	46.95%	38.00%	27.29%	16.91%	8.54%
L_4	49.87%	42.39%	32.94%	21.69%	14.21%
L_5	52.80%	45.57%	36.82%	28.87%	18.76%

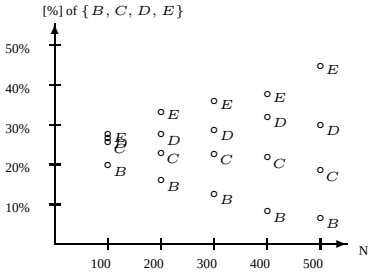
	$\mathbb{R}^5$	$\mathbb{R}^6$	$\mathbb{R}^7$	$\mathbb{R}^8$	$\mathbb{R}^9$
L_1	31.70%	19.12%	8.04%	2.65%	0.39%
L_2	31.57%	22.93%	13.88%	5.97%	1.68%
L_3	36.27%	25.48%	17.11%	10.44%	3.30%
L_4	37.72%	27.53%	19.37%	10.61%	3.83%
L_5	39.22%	29.81%	21.48%	14.85%	4.81%

Figure 3: Recognition of $S_{L_1L_2L_3L_4L_5} \sharp t^x \approx L_x$ in HRR and GA$_c$ using clean-up memory containing letters only, 1000 trials.

t are different each time the black box produces a sequence. We will use an algorithm based on the assumption that t^x, if not recognized correctly, is more similar to highest powers of t as shown in Section 4. The second assumption is that letters lying closer to the end of the sequence are often offered as the incorrect answer to questions concerning letters (Section 5). The clean-up memory $\mathcal{C}$ for this experiment consists of all five letters and the five powers of t. We will also use an auxiliary clean-up memory $\mathcal{L}$ containing letters only.

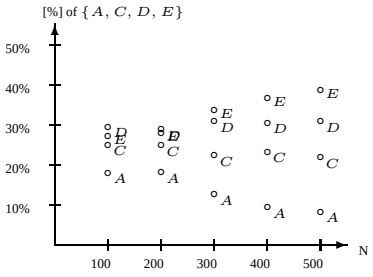
The algorithm for finding out the position of each letter begins with asking a question described by equation (1) — "Where in the sequence $S_{\bullet\bullet\bullet\bullet\bullet}$ is the letter L_x?":

$$S_{\bullet\bullet\bullet\bullet\bullet} \sharp L_x = \left\{ \begin{array}{ll} S_{\bullet\bullet\bullet\bullet\bullet} \circledast (L_x)^* & \text{in HRR} \\ (L_x)^+ S_{\bullet\bullet\bullet\bullet\bullet} & \text{in GA}_c \end{array} \right\} = (t^x)' \approx t^x \quad (1)$$



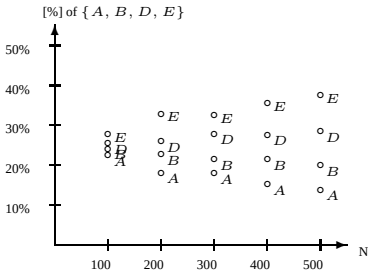
Asking with A

	N=100	N=200	N=300	N=400	N=500	
non A	1643	960	616	506	393	= 100%
B	19.90%	16.04%	12.66%	8.30%	6.62%	
C	25.56%	22.92%	22.73%	21.94%	18.58%	
D	26.78%	27.71%	28.73%	32.02%	30.03%	
E	27.75%	33.33%	35.88%	37.75%	44.78%	



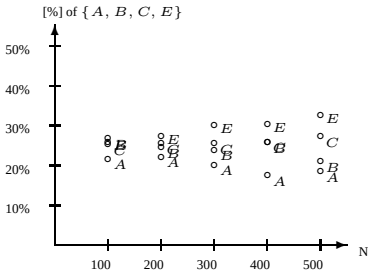
Asking with B

	N=100*	N=200*	N=300	N=400	N=500	
non B	2474	1588	1133	837	628	= 100%
A	18.03%	18.14%	12.71%	9.44%	8.28%	
C	25.14%	24.87%	22.51%	23.30%	21.97%	
D	29.59%	28.90%	30.98%	30.59%	30.89%	
E	27.24%	28.09%	33.80%	36.68%	38.85%	



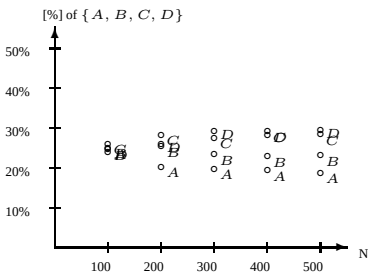
Asking with C

	N=100	N=200	N=300	N=400	N=500	
non C	3003	2214	1694	1346	1121	= 100%
A	22.51%	18.16%	18.00%	15.30%	13.83%	
B	24.18%	22.81%	21.55%	21.47%	20.07%	
D	25.51%	26.06%	27.86%	27.64%	28.55%	
E	27.81%	32.97%	32.59%	35.59%	37.56%	



Asking with D

	N=100*	N=200	N=300	N=400	N=500	
non D	3432	2686	2264	1808	1564	= 100%
A	21.71%	22.23%	20.19%	17.64%	18.67%	
B	26.84%	24.72%	24.03%	25.77%	21.16%	
C	25.44%	25.58%	25.66%	26.05%	27.37%	
E	26.02%	27.48%	30.12%	30.53%	32.80%	



Asking with E

	N=100*	N=200*	N=300	N=400*	N=500	
non E	3754	3116	2617	2353	1921	= 100%
A	24.77%	20.31%	19.79%	19.46%	18.84%	
B	25.12%	25.48%	23.42%	22.90%	23.27%	
C	26.08%	28.34%	27.59%	29.20%	28.37%	
D	24.03%	25.87%	29.19%	28.35%	29.52%	

Figure 4: Finding letter alignment in a sequence S_{ABCDE} in HRR, 10000 trials.

for each letter $L_x \in \mathcal{L}$. Next, we need to find the item in the clean-up memory $\mathcal{C} \setminus \mathcal{L}$ that is most similar to $(t^x)'$. Let us denote this item by z. With high probability, z is the power of t associated with the position of the letter L_x in the sequence $S_{\bullet\bullet\bullet\bullet\bullet}$, although, if recognized incorrectly, z will most likely point to some other $t^{y>x}$. Now let us ask a second question (eq. (2)) — "Which letter is situated at the z'th position in the sequence $S_{\bullet\bullet\bullet\bullet\bullet}$?":

$$S_{\bullet\bullet\bullet\bullet\bullet} \sharp z = \left\{ \begin{array}{ll} S_{\bullet\bullet\bullet\bullet\bullet} \circledast z^* & \text{in HRR} \\ \langle S_{\bullet\bullet\bullet\bullet\bullet}\, z^+ \rangle_1 & \text{in GA}_c \end{array} \right\} = L' \approx L_x. \quad (2)$$

We use the projected product in GA_c because we are looking for a letter vector placed on the position indicated by

z. In HRR the resulting L' should be compared with letters only. In most cases L' will point to the correct letter. However, in a small fraction of test results, L' will point to letters surrounding L_x, because z has been mistakenly decoded as t^y for some $y \neq x$. Also, letters preceding L_x should come up less often than letters proceeding L_x.

Figure 4 presents test results for HRR. The data in Figure 4 should be interpreted as follows: the first row of each table next to a graph contains the vector lengths of the data used in 5 consecutive experiments (10000 trials each). The second row contains the number of faulty answers within those 10000 trials. The next 4 rows present the percentage of occurence of a "faulty" letter within all faulty answers presented in the second row.

Faulty alignments (i.e. those, for which the percentages corressponding to letters do no align increasingly within a single column) have been marked with a "*" in the table headings. We used S_{ABCDE} as the mysterious encoded sequence $S_{\bullet\bullet\bullet\bullet\bullet}$. In each case we crossed out the most frequently occurring letter and we concentrated on the frequency of the remaining letters. In HRR, for sufficiently large vector sizes, the frequencies f_L of all letters $L \in \mathcal{L}$ aligned correctly

$$\begin{aligned}
f_B < f_C < f_D < f_E \qquad & \text{asking with } A,\\
f_A < f_C < f_D < f_E \qquad & \text{asking with } B,\\
f_A < f_B < f_D < f_E \qquad & \text{asking with } C,\\
f_A < f_B < f_C < f_E \qquad & \text{asking with } D,\\
f_A < f_B < f_C < f_D \qquad & \text{asking with } E.
\end{aligned}$$

It was straightforward that these inequalities lead to $f_A < f_B < f_C < f_D < f_E$ and correctly identify the encoded sequence as S_{ABCDE}. Test results are less accurate when we asked about letters lying closer to the end of a sequence, therefore the size of the vector should be adequately long. Moreover, the longer the vector, the larger the difference between the frequencies.

GA_c was expected to perform worse in this experiment, because we can construct powers of a mulivector t^{i-1} by multiplying it with t from one side only. Another reason for poor performance was that the frequencies of the powers of t recognized incorrectly tend to cluster in GA_c (Figure 2). Indeed, Table 1 shows that letter frequencies do not align correctly at all. We therefore needed to slightly modify the algorithm for finding letter alignment in GA_c. Since powers of t are more similar to each other in GA_c than in HRR (Section 5), we concentrated on two largest frequencies in each series of asking questions — the largest frequency represents the letter L that was used to ask the question and the second largest frequency indicates letter $\hat{L}$ that most likely proceeds letter L.

Table 1 presents the frequencies of letters recognized as the most probable answer to Equation (2), the second largest frequency in each row is printed in bold. Partial letter alignments have been placed next to each table and contradictory alignments have been preceded with a "*". When being asked with the last letter of the sequence, HRR provided less accurate answers and so did GA_c by yielding more contradictions than in case of previous letters. It is impossible to avoid contradictory alignments in GA_c because we do not know which letter is the last one and the algorithm for recovering letter alignment in GA_c instructs us to write down the partial alignment with that letter being proceeded by another letter. The remaining alignments point correctly to the sequence S_{ABCDE}

$$\left.\begin{array}{c} A \prec B \\ C \prec D \prec E \\ A \prec B \prec C \prec D \\ A \prec C \\ B \prec D \prec E \\ A \prec E \end{array}\right\} \Rightarrow A \prec B \prec C \prec D \prec E.$$

7 Conclusion

We have shown that multivector powers in GA_c have properties similar to convolutive powers of HRR vectors

- (multi)vectors t^{i-r} and t^i are similar in much the same way as t^i and t^{i+r},
- items placed near the beginning of a sequence are remembered more prominently and thus, are recognized correctly more often,
- items placed near the end of a sequence are remembered less precisely and often come up as the most probable answer when the correct item is not recognized.

We have used the last two properties to find the alignment of sequence items without the explicit knowledge of (multi)vector powers. While HRR retrieved the original alignment without greater problems, GA_c left us with an easily soluble logical puzzle providing fragmentary alignments.

These properties can be used to build holographic lexicons, dictionaries and other structures that require storing order information and word meaning in the same pattern.

Acknowledgement

This work was supported by *Grant G.0405.08* of the *Fund for Scientific Research Flanders*.

References

[1] D. Aerts and M. Czachor (2004), "Quantum aspects of semantic analysis and symbolic artificial intelligence", *J. Phys. A*, vol. 37, pp. L123-L13.

[2] D. Aerts and M. Czachor (2007), "Cartoon computation: Quantum-like algorithms without quantum mechanics", *J. Phys. A*, vol. 40, pp. F259-F266.

Table 1: Finding letter alignment in a sequence S_{ABCDE} in GA$_c$, 10000 trials.

$\mathbb{R}^5$	f_A	f_B	f_C	f_D	f_E
asking with A	71.60%	**8.22**%	6.44%	2.72%	6.47%
asking with B	8.98%	65.92%	9.16%	6.76%	**9.18**%
asking with C	7.28%	9.52%	67.25%	**9.67**%	6.28%
asking with D	8.74%	6.75%	8.80%	66.83%	**8.88**%
asking with E	8.03%	**9.96**%	6.83%	9.28%	65.90%

$A \prec B$, $^*B \prec E$, $C \prec D$, $D \prec E$, $^*E \prec B$ — $A \prec B$, $C \prec D \prec E$

$\mathbb{R}^6$	f_A	f_B	f_C	f_D	f_E
asking with A	80.34%	**5.34**%	4.61%	5.08%	4.63%
asking with B	6.56%	73.91%	**7.48**%	4.67%	7.38%
asking with C	6.71%	7.47%	72.83%	**7.68**%	5.31%
asking with D	6.79%	5.37%	7.42%	72.30%	**8.12**%
asking with E	5.52%	7.77%	5.58%	**8.54**%	72.59%

$A \prec B$, $B \prec C$, $C \prec D$, $^*D \prec E$, $^*E \prec D$ — $A \prec B \prec C \prec D$

$\mathbb{R}^7$	f_A	f_B	f_C	f_D	f_E
asking with A	89.78%	2.42%	**2.91**%	2.37%	2.52%
asking with B	3.78%	83.92%	4.04%	**4.44**%	3.82%
asking with C	4.30%	4.79%	80.54%	5.11%	**5.26**%
asking with D	4.35%	5.07%	5.41%	79.09%	**6.08**%
asking with E	4.57%	5.07%	**5.85**%	5.68%	78.83%

$A \prec C$, $B \prec D$, $^*C \prec E$, $D \prec E$, $^*E \prec C$ — $A \prec C$, $B \prec D \prec E$

$\mathbb{R}^8$	f_A	f_B	f_C	f_D	f_E
asking with A	95.33%	0.88%	1.26%	1.20%	**1.30**%
asking with B	1.34%	92.27%	1.72%	**2.93**%	1.74%
asking with C	2.15%	2.28%	88.99%	2.35%	**4.23**%
asking with D	1.93%	**3.75**%	2.82%	88.19%	3.31%
asking with E	2.60%	2.36%	**5.09**%	3.32%	86.63%

$A \prec E$, $^*B \prec D$, $^*C \prec E$, $^*D \prec B$, $^*E \prec C$ — $A \prec E$

[3] M. Czachor (2007), “Elementary gates for cartoon computation", *J. Phys. A*, vol. 40, pp. F753-F759.

[4] D. Aerts and M. Czachor (2008), “Tensor-product versus geometric-product coding", *Physical Review A*, vol. 77, id. 012316.

[5] D. Aerts, M. Czachor, and B. De Moor (2009), “Geometric Analogue of Holographic Reduced Representation", *J. Math. Psychology*, vol. 53, pp. 389-398.

[6] D. Aerts, M. Czachor, and B. De Moor (2006), “On geometric-algebra representation of binary spatter codes". preprint arXiv:cs/0610075 [cs.AI].

[7] D. Aerts, M. Czachor, and Ł. Orłowski (2009), “Teleportation of geometric structures in 3D ", *J. Phys. A* vol. 42, 135307.

[8] W.K. Clifford (1878), “Applications of Grassmann’s extensive algebra", *American Journal of Mathematics Pure and Applied*, vol. 1, 350–358.

[9] R. W. Gayler (1998), “Multiplicative binding, representation operators, and analogy", *Advances in Analogy Research: Integration of Theory and Data from the Cognitive, Computational, and Neural Sciences*, K. Holoyak, D. Gentner, and B. Kokinov, eds., Sofia, Bulgaria: New Bulgarian University, p. 405.

[10] S. Deerwester et al. (1990), “Indexing by Latent Semantic Analysis", *Journal of American Society for Information Science*, vol. 41, 391.

[11] H. Grassmann (1877), “Der Ort der Hamilton’schen Quaternionen in der Ausdehnungslehre”, *Mathematische Annalen*, vol. 3, 375–386.

[12] M.N. Jones & D.J.K. Mewhort (2007), “Representing Word Meaning and Order Information in a Composite Holographic Lexicon", *Psychological Review*, vol. 114, No. 1, pp. 1-37.

[13] G. E. Hinton, J. L. McClelland and D. E. Rumelhart (1986), “"Parallel distributed processing: Explorations in the microstructure of cognition", vol. 1, 77Ű109, *"Distributed representations"*, The MIT Press, Cambridge, MA.

[14] P. Kanerva (1996), “Binary spatter codes of ordered k-tuples”. In C. von der Malsburg et al. (Eds.), *Artificial Neural Networks ICANN Proceedings, Lecture Notes in Computer Science* vol. 1112, pp. 869-873.

[15] P. Kanerva (1997), “Fully distributed representation”. *Proc. 1997 Real World Computing Symposium (RWCŠ97, Tokyo)*, pp. 358-365.

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

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

[18] N.G. Marchuk, and D.S. Shirokov (2008), "Unitary spaces on Clifford algebras", *Advances in Applied Clifford Algebras*, vol 18, pp. 237-254.

[19] A. Patyk (2010), "Geometric Algebra Model of Distributed Representations", in *Geometric Algebra Computing in Engineering and Computer Science*, E. Bayro-Corrochano and G. Scheuermann, eds. Berlin: Springer. Preprint arXiv:1003.5899v1 [cs.AI].

[20] A. Patyk-Łońska (2011), "Distributed Representations Based on Geometric Algebra: the Continuous Model", submitted to *Informatica*.

[21] R. Pike (1984), "Comparison of Convolution and Matrix Distributed Memory Systems for Associative Recall and Recognition", *Psychological Review*, vol. 91, No. 3, pp. 281-294.

[22] T. Plate (1995), "Holographic Reduced Representations", *IEEE Trans. Neural Networks*, vol. 6, no. 3, pp. 623-641.

[23] T. Plate (2003), *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CSLI Publications, Stanford.

[24] D.A. Rachkovskij (2001), "Representation and Processing of Structures with Binary Sparse Distributed Codes", *IEEE Trans. Knowledge Data Engineering*, vol. 13, no. 2, pp. 261-276.

[25] P. Smolensky (1990), "Tensor product variable binding and the representation of symbolic structures in connectionist systems". *Artificial Intelligence*, vol. 46, pp. 159-216.