Knowl Inf Syst
DOI 10.1007/s10115-016-1012-2

REGULAR PAPER

Random indexing of multidimensional data

Fredrik Sandin[1] · Blerim Emruli[2] ·
Magnus Sahlgren[3]

Received: 3 September 2014 / Revised: 30 May 2016 / Accepted: 18 November 2016

Abstract Random indexing (RI) is a lightweight dimension reduction method, which is used, for example, to approximate vector semantic relationships in online natural language processing systems. Here we generalise RI to multidimensional arrays and therefore enable approximation of higher-order statistical relationships in data. The generalised method is a sparse implementation of random projections, which is the theoretical basis also for ordinary RI and other randomisation approaches to dimensionality reduction and data representation. We present numerical experiments which demonstrate that a multidimensional generalisation of RI is feasible, including comparisons with ordinary RI and principal component analysis. The RI method is well suited for online processing of data streams because relationship weights can be updated incrementally in a fixed-size distributed representation, and inner products can be approximated on the fly at low computational cost. An open source implementation of generalised RI is provided.

Keywords Data mining · Random embeddings · Dimensionality reduction · Sparse coding · Semantic similarity · Streaming algorithm · Natural language processing

1 Introduction

There is a rapid increase in the annual amount of data that is produced in almost all domains of science, industry, economy, medicine and even everyday life. We have surpassed a critical point where more data are generated than we can physically store. Choosing which data to archive and process and which to discard is necessary in data-intensive applications.

✉ Fredrik Sandin
fredrik.sandin@ltu.se

1 EISLAB, Luleå University of Technology, 971 87 Luleå, Sweden

2 SICS Swedish ICT, 722 13 Västerås, Sweden

3 SICS Swedish ICT, 164 29 Kista, Sweden

That trend motivates the development of new methods for data representation and analysis [2,23,49].

One interesting approach to analyse large data sets is to search for outstanding relationships between "features" in the data. Problems of that type naturally appear in the form of context- or time-dependent relationships. For example, the co-occurrence of words in articles, blogs and so on is one type of relationship that carries information about language use and evolution over time [46]. Similarly, co-occurrence analysis can be used to investigate the general opinion about things, for example public events or politicians, and how the opinion changes over time [51]. The analysis requires averaging over many instances of relationships in order to identify frequent or otherwise significant patterns in a noise-like background. That is a non-trivial problem because the number of possible relationships between elements of the sets A_i scales like $\mathcal{O}(\Pi_i |A_i|)$, where $|A_i|$ denotes the cardinality of the set A_i. In the example of online text analysis $|A| \sim 10^5$, which implies $\sim 10^{10}$ co-occurrence weights that evolve over time and typically depend on additional context variables of interest. Therefore, the number of relationship weights that need to be stored and updated in such applications can be astronomical, and the analysis prohibitive given the large size of the data representation.

This is the motivation of random indexing (RI) [31], which is a random-projection method that solves such problems by incrementally generating distributional representations that approximate similarities in sets of co-occurrence weights. For example, in the context of natural language processing the RI method is used to compress large word–document or word–context co-occurrence matrices [52]. This is done by associating each document or context with a sparse random ternary vector of high dimensionality [29,30], a so-called index vector. Each word is also represented by a high-dimensional vector of integers, a so-called distributional vector. These distributional vectors are initially set to zero, and for each appearance of a particular word in a context, the index vector of that context is added to the distributional vector of the word. The result of this incremental process is that words that appear in similar contexts get similar distributional vectors, indicating that they are semantically related [44]. Therefore, the analysis of semantic similarity can be performed by comparing the compressed distributional vectors in terms of inner products, instead of analysing and storing the full co-occurrence matrix. The distributional vectors can be updated on the fly in streaming applications by adding the appropriate sparse index vectors and co-occurrence weights to the distributional vectors. See Sahlgren [42,43] for further details.

LSA [14] and HAL [36] are two other prominent examples of vector-space models [52] used for semantic analysis of text. In these methods, a co-occurrence matrix is explicitly constructed, and then singular value decomposition (SVD) is used to identify the semantic relationships between terms (see [8] for recent examples). This process requires significant storage space for the full co-occurrence matrix, and it is a computationally costly method. The SVD can be calculated using parallel and iterative methods optimised for sparse matrices [4], but the computational cost still prevents the processing of large and streaming data sets [11]. In contrast, RI easily scales to large corpora such as the MEDLINE collection of approximately 9 million abstracts [10]. Another approach known as locality-sensitive hashing (LSH) [7] is compared with RI on a distributional similarity task by Gorman and Curran [21], showing that RI outperforms LSH in terms of efficiency and accuracy when the problem size increases. RI requires a fraction of the memory and processing power of LSA and HAL [11], but is comparable with models based on SVD in terms of accuracy. For example, the accuracy of RI is comparable to SVD-based methods in a TOEFL synonym identification task [31], and that result has been further improved in the case of RI [45]. RI of co-occurrence matrices for semantic analysis works surprisingly well [11,30,43,52], and the method has been adopted in other applications, such as indexing of literature databases [54], event detection in blogs [26], web user clustering and page prefetching [57], graph searching for the semantic web [12], diagnosis code assignment to patients [24], predicting speculation in biomedical literature [55] and failure prediction [19]. In general, there is an increasing interest for randomisation in information processing because it enables the use of simple algorithms, which can be organised to exploit parallel computation in an efficient way [6,22].

The practical usefulness of RI is also demonstrated by several implementations in public software packages such as the S-Space Package [27] and the Semantic Vectors Package [58], and extensions of the basic method to new domains and problems [26,54]. Therefore, it is natural to ask whether the RI algorithm can be generalised to higher-order relationships and distributional arrays.

In the next section we generalise RI of vectors to RI of multidimensional data in the form of matrices and higher-order arrays. Subsequently, we present results of simulation experiments of ordinary and generalised RI demonstrating some properties of the generalised method, including a comparison with principal component analysis (PCA). PCA and similar approximation methods for higher-order arrays such as Tucker decomposition [34] are expected to result in higher signal-to-noise ratio (SNR) than RI when applicable because the dimension reduction is optimised to minimise the residual variance. However, such methods are more complex and target another application domain. We conclude that the possibility to incrementally encode and analyse general co-occurrence relationships at low computational cost using a distributed representation of approximately fixed size makes generalised RI interesting for online processing of data streams.

2 Method

In ordinary RI [31,42], the index vectors $\mathbf{r}(x_j)$ are used to calculate distributional vectors $\mathbf{s}(x_i)$ by adding the index vectors of the context items x_j to the distributional vector of word x_i every time that word occurs in the data. This can be formalised as

$$\mathbf{s}(x_i) \leftarrow \mathbf{s}(x_i) + \sum_{j=-c, j\neq 0}^{c} w(x_{i+j})\pi_j \mathbf{r}(x_{i+j}), \tag{1}$$

where c is the number of items surrounding a word that defines the context window, $w(x_j)$ is a weight function that quantifies the importance of a context item x_j, and π_j is an optional permutation operator that makes the context window word-order dependent [45]. This way RI can, for example, be used to identify words that appear in similar contexts by analysing the inner products of the distributional vectors, $\mathbf{s}$, thereby greatly simplifying the co-occurrence analysis problem outlined above.

In the following we refer to RI of vectors as one-way RI and generalise one-way RI to n-way RI of arrays $a_{i_1,i_2,i_3,\ldots,i_\mathcal{N}}$ of arbitrary order, meaning that there are n sets of index vectors associated with each dimension of the array. We focus on the core RI mechanism and omit extensions like the word-order-dependent permutation introduced above in order to make the presentation more accessible. Array elements are denoted with $a_{i_1,i_2,i_3,\ldots,i_\mathcal{N}}$, or $a_{\bar{i}}$ for short, and the indices $\{i_1, i_2, i_3, \ldots, i_\mathcal{N}\}$ are used in array element space. The array elements are encoded in a distributed fashion in *states* that are denoted with $s_{\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_\mathcal{N}}$, or $s_{\bar{\alpha}}$ for short. The indices $\{\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_\mathcal{N}\}$ are used in state space. We use the notation $i_\mathcal{D}$ when referring to indices of the array space and $\alpha_\mathcal{D}$ when referring to indices of the state space, where $\mathcal{D}$ is the dimension index. For vectors $\mathcal{D} = 1$, for matrices $\mathcal{D} \in \{1, 2\}$ and in general $\mathcal{D} \in [1, \mathcal{N}]$. When necessary we use one additional index, $j_{\mathcal{D}}$, in array element space. Similarly, one additional state-space index, $\beta_{\mathcal{D}}$, is used when necessary.

States have physical representations that are stored in memory, but they are only accessed using particular decoder and encoder functions (introduced below) which generalise (1) for vectors to arrays of arbitrary order. The array elements are related to the states by a random projection [56] mechanism and constitute the input to the encoder function and the output from the decoder function, respectively. The order of the state array, $\mathcal{N}$, is equivalent to that of the array. The core idea is that the state array can be of significantly smaller size than the array itself and that approximate vector semantic analysis can be performed in state space at low computational cost. Similarly, the set of distributional vectors, $\mathbf{s}$, in (1) have few elements compared to the full co-occurrence matrix. This possibility follows from the Johnson–Lindenstrauss lemma [25], which describes the mapping of points in a high-dimensional space to a space of lower dimension so that the distances between points are approximately preserved. Further developments of the Johnson–Lindenstrauss lemma and related applications can be found, for example, in [1,13,18,28,38].

2.1 Random indexing

For each index of the array, $i_{\mathcal{D}}$, there is an associated random-index array, $r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}}$. If $\mathcal{D}$ and $i_{\mathcal{D}}$ are fixed, the state-space elements of $r_{\mathcal{D},i_{\mathcal{D}},:}$ form a sparse high-dimensional ternary vector, a so-called *index vector*

$$r_{\mathcal{D},i_{\mathcal{D}},:} = [\ldots\ 0\,0\,0\,1\,0\,0\,0\ \ldots\ 0\,0\,0\,{-1}\,0\,0\,0\ldots]_{\mathcal{D},i_{\mathcal{D}}}. \tag{2}$$

Index vectors have a few nonzero elements at random positions $\alpha_{\mathcal{D}}$, hence the name "random index". The nonzero elements of an index vector have an absolute value of one, and half of these values are negative. In other words, index vectors are sparse ternary vectors with elements called *trits*. This definition simplifies to ordinary RI, $\mathbf{r}(x_j)$ in (1), in the case of $\mathcal{N} = 1$.

The number of nonzero trits in the index vectors, $\chi_{\mathcal{D}}$, is a model parameter that typically has a value of order ten [42]. Therefore, as we explain in the following sections, each index vector defines a random projection on a sparse subset of the states. We denote the ranges of state indices, $\alpha_{\mathcal{D}}$, with $[1, L_{\mathcal{D}}]$ so that, for example, $\alpha_1 \in [1, L_1]$ and $\alpha_2 \in [1, L_2]$. Similarly, the ranges of the element indices, $i_{\mathcal{D}}$, are $[1, N_{\mathcal{D}}]$. The length of an index vector is equivalent to the maximum value of the state index, $L_{\mathcal{D}}$, in each dimension. For example, if the state array of a matrix is of size 1000×2000 the index vectors would be of length 1000 for $\mathcal{D} = 1$ and 2000 for $\mathcal{D} = 2$, respectively. Index vectors can be represented in compact form because most of the elements are zero. Here the indices of the nonzero trits are used to represent the index vectors, and the signs are implicitly encoded with the position of the indices so that the first half of the list of indices are associated with positive signs. The number of nonzero trits in an index vector, $\chi_{\mathcal{D}}$, is an even number. For each dimension, $\mathcal{D}$, there are $N_{\mathcal{D}}$ index vectors of length $L_{\mathcal{D}}$, and each index vector has $\chi_{\mathcal{D}}$ nonzero trits. In practical applications, an index vector is represented in compact form by at most a few dozen integers. Therefore, the storage space required for higher-order RI representations is practically determined by the size of the state array. A summary of parameters and their definitions is presented in Table 1.

The notation and definitions introduced above are a direct generalisation of ordinary RI to arrays of arbitrary order. In particular, ordinary RI is defined by $\mathcal{N} = 1$. Note that the states defined here correspond to the elements of the distributional vectors in ordinary RI, which are the hard storage locations where the distributional statistics are stored. Next we

Table 1 Summary of parameters

Expression	Description
$a_{i_1,i_2,i_3,\ldots,i_\mathcal{N}}$, $a_{\bar{i}}$	Array elements
$s_{\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_\mathcal{N}}$, $s_{\bar{\alpha}}$	State array, accessed by encoder/decoder functions
$\mathcal{N}$	Dimensionality of array
$\mathcal{D}$	Dimension index, $1 \leq \mathcal{D} \leq \mathcal{N}$
$N_\mathcal{D}$	Number of index vectors in dimension $\mathcal{D}$, $i_\mathcal{D} \in [1, N_\mathcal{D}]$
$L_\mathcal{D}$	Length of index vectors in dimension $\mathcal{D}$, $\alpha_\mathcal{D} \in [1, L_\mathcal{D}]$
$\chi_\mathcal{D}$	Number of nonzero trits in index vectors of dimension $\mathcal{D}$
$S_e = \prod_\mathcal{D} \chi_\mathcal{D}$	Number of states that encode one array element
$S_s \propto \prod_\mathcal{D} L_\mathcal{D}$	Disk/memory space required to store the state array
$S_r \propto \sum_\mathcal{D} N_\mathcal{D} \chi_\mathcal{D}$	Disk/memory space required to store index vectors

present the corresponding generalised encoding algorithm and generalised method for vector semantic analysis in terms of inner products.

2.2 Encoding algorithm

The states, $s_{\bar{\alpha}}$, are initially set to zero ($s_{\bar{\alpha}}^{t=0} = 0$), which implies that the array elements, $a_{\bar{i}}$, are zero also (see Sect. 2.3 for details). In a typical application of RI, the array elements are incrementally updated, for example, by adding co-occurrence weights derived from streaming text to the array elements in a cumulative manner like in (1). An array element, $a_{\bar{i}}$, is encoded in the state array, $s_{\bar{\alpha}}$, in a sparse and distributed fashion using a random projection defined by the product of index vectors. Addition of a scalar weight, $w_{i_1,i_2,i_3,\ldots,i_\mathcal{N}}$, to a particular array element, $a_{\bar{i}}$, is defined by

$$s_{\bar{\alpha}}^{t} = s_{\bar{\alpha}}^{t-1} + w_{\bar{i}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_\mathcal{D},\alpha_\mathcal{D}}, \tag{3}$$

where the indices $\bar{i}$ are determined by the choice of array element and $s_{\bar{\alpha}}^{t}$ ($s_{\bar{\alpha}}^{t-1}$) denotes the resulting (current) state array. By applying (3) in an iterative fashion the weights of array elements can be *incrementally* updated, without modifying the random projections or recalculating substantial parts of the state array. Furthermore, this definition implies that the indices of an array element are used to select a particular set of index vectors, forming an outer product of "nearly orthogonal", or so-called *indifferent* index vectors in state space (see "Appendix" for further details).

The outer product of index vectors in (3) is a sparse array that has S_e nonzero elements with values of either $+1$ or -1, where

$$S_e = \prod_{\mathcal{D}=1}^{\mathcal{N}} \chi_\mathcal{D}. \tag{4}$$

The computational cost of the encoding algorithm is proportional to S_e, which is constant. Therefore, the encoding complexity for an input sequence of length n is $\mathcal{O}(n)$ for RI of any order, which is lower than the complexity of streaming PCA [39]. Furthermore, new array elements (relationship weights) can be added to the representation with low impact on the representation size; see the discussion in Sect. 2.1. These two properties make the generalised RI algorithm interesting for streaming data applications.

Subtraction of $w_{\bar{i}}$ is defined by the replacement $w_{\bar{i}} \to -w_{\bar{i}}$ in (3). Assignment of array elements is not defined because of the distributional nature of the representation of array elements.

2.3 Decoding algorithm

Vector semantic analysis can be performed in state space, without the need to first decode array elements. This is key because decoding and explicit processing of array elements are a computationally costly operation. It is anyway instructive to outline a generalised decoding procedure for multidimensional RI. The decoding operation is a projection of the state array on the index vectors that correspond to each particular array element

$$a_{\bar{i}} = \frac{1}{S_e} \sum_{\alpha_1=1}^{L_1} \sum_{\alpha_2=1}^{L_2} \sum_{\alpha_3=1}^{L_3} \cdots \sum_{\alpha_{\mathcal{N}}=1}^{L_{\mathcal{N}}} s_{\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_{\mathcal{N}}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}}, \tag{5}$$

where S_e^{-1} is a normalisation factor defined above that compensates for the redundancy of the distributed representation of $a_{\bar{i}}$ in the states, $s_{\bar{\alpha}}$. The complexity of this algorithm is comparable to that of the encoding algorithm outlined above because S_e different states are processed in both cases.

The encoding procedure (3) is a sequence of outer products of indifferent index vectors, and the decoding procedure is the corresponding sequence of inner products. It follows from (3) and (5) that the decoded value is an exact reconstruction of the accumulated encoded weight if all index vectors are orthogonal. However, that process would be useless in the context considered here because no dimension reduction is achieved in that case. For index vectors of length $L_{\mathcal{D}}$, at most $L_{\mathcal{D}}$ linearly independent vectors can be constructed (a set of basis vectors). For high values of $L_{\mathcal{D}}$ there are many more vectors that are approximately orthogonal; see "Appendix" for details, which makes it possible to encode and decode approximate array elements in a small state space provided that the data are sufficiently sparse (see Sect. 3 for details).

Combining the encoding operation (3) and the decoding operation (5) the following expression results for the decoded weight, $\tilde{w}_{\bar{i}}$, of an array element

$$\begin{aligned}
\tilde{w}_{\bar{i}} &= \frac{1}{S_e} \sum_{\alpha_1=1}^{L_1} \sum_{\alpha_2=1}^{L_2} \sum_{\alpha_3=1}^{L_3} \cdots \sum_{\alpha_{\mathcal{N}}=1}^{L_{\mathcal{N}}} \left(\epsilon_{\bar{\alpha}} + w_{\bar{i}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} \right) \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} && (6)\\
&= \frac{1}{S_e} \sum_{\bar{\alpha}} \left(\epsilon_{\bar{\alpha}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} + w_{\bar{i}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r^2_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} \right) && (7)\\
&= \epsilon_{\bar{i}} + \frac{w_{\bar{i}}}{S_e} \prod_{\mathcal{D}=1}^{\mathcal{N}} \chi_{\mathcal{D}} && (8)\\
&= \epsilon_{\bar{i}} + w_{\bar{i}}. && (9)
\end{aligned}$$

Here $\epsilon_{\bar{\alpha}}$ indicates the noise associated with the distributed coding of the weights and the non-orthogonality of the index vectors, resulting in an absolute error $\epsilon_{\bar{i}} = \tilde{w}_{\bar{i}} - w_{\bar{i}}$ of decoded weights. We return to the discussion of the error term, $\epsilon_{\bar{i}}$, in the next section, which presents simulation results. Partial results that may be helpful to derive analytical bounds on $\epsilon_{\bar{i}}$ are included in "Appendix".

2.4 Generalised vector semantic analysis

The RI method that is used in natural language processing is based on distributional vectors [30,31,42,43]. Each term that appears in a text corpus is associated with a distributional vector, and each context or document is associated with a ternary index vector; see (1). Therefore, a distributional vector corresponds to the states of a one-dimensional RI array, $\mathcal{N} = 1$, and the conventional index vectors correspond to the ternary index vectors of that array. The definition of the encoding operation (3) reduces to ordinary RI (1) in the case of one-way RI of vectors and constitutes a natural generalisation of RI to higher-order arrays.

A central aspect of RI is that the distributional vectors, $\mathbf{s}$, which are aggregated representations of encoded semantic relationships, can be used for direct comparison of semantic similarity using, for example, an inner-product, $\mathbf{s}_1 \cdot \mathbf{s}_2$, or cosine-of-angle measure. That approach is similar to the seminal works by Papadimitriou et al. [41], Kaski [33] and others [5,17], which are motivated by the Johnson–Lindenstrauss lemma [25]. In the case of generalised RI developed here, a similar method can be derived from (5) by considering the following inner product of decoded weight-vectors,

$$\sum_{i_1=j_1=1}^{N_1} a_{\bar{i}} a_{\bar{j}} = S_e^{-2} \sum_{i_1=j_1=1}^{N_1} \left(\sum_{\bar{\alpha}} s_{\bar{\alpha}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} \right) \left(\sum_{\bar{\beta}} s_{\bar{\beta}} \prod_{\mathcal{D}=1}^{\mathcal{N}} r_{\mathcal{D},j_{\mathcal{D}},\beta_{\mathcal{D}}} \right) \tag{10}$$

$$= S_e^{-2} \sum_{k=1}^{N_1} \left(\sum_{\alpha_1=1}^{L_1} r_{1,k,\alpha_1} \sum_{\alpha_2=1}^{L_2} \sum_{\alpha_3=1}^{L_3} \cdots \sum_{\alpha_{\mathcal{N}}=1}^{L_{\mathcal{N}}} s_{\bar{\alpha}} \prod_{\mathcal{D}=2}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} \right)$$

$$\times \left(\sum_{\beta_1=1}^{L_1} r_{1,k,\beta_1} \sum_{\beta_2=1}^{L_2} \sum_{\beta_3=1}^{L_3} \cdots \sum_{\beta_{\mathcal{N}}=1}^{L_{\mathcal{N}}} s_{\bar{\beta}} \prod_{\mathcal{D}=2}^{\mathcal{N}} r_{\mathcal{D},j_{\mathcal{D}},\beta_{\mathcal{D}}} \right), \tag{11}$$

where the indices $\{i_2, i_3 \ldots i_{\mathcal{N}}\}$ and $\{j_2, j_3 \ldots j_{\mathcal{N}}\}$ are constant and specify one particular inner product (one relationship). The sums over $\bar{\alpha}$ and $\bar{\beta}$ have relatively few nonzero terms due to the sparse structure of the index vectors. The sum over $i_1 = j_1 = k$ has many terms and needs to be approximated in order to reduce the computational complexity of the method, which is necessary in order to enable large-scale explorative studies of semantic similarity.

The vectors $r_{1,k,:}$ in (11) are sparse ternary vectors defined by (2) that maps each value of k to multiple values of α_1 and β_1 for which there are nonzero contributions from the sum over states to the inner product. More specifically, the number of such values for α_1 and β_1 is exactly χ_1 for each value of k, which implies that there are χ_1 nonzero "terms" in each of the sums over α_1 and β_1. Therefore, the explicit evaluation of the inner product involves pseudorandom sampling of state indices α_1 and β_1, which can be approximated with an explicit sum over all possible values of these state indices. Therefore, the number of terms in the sums over k and α_1 (and β_1) is reduced from $\chi_1 N_1$ to L_1, which is a significant improvement. This is analogous to ordinary RI, where the distributional vectors, $\mathbf{s}$ in (1), are compared for similarity directly, without prior decoding (inverse random projection) of word–context co-occurrence weights. Furthermore, the accuracy of the approximation can be improved by averaging the states selected by the constant indices $\{i_2, i_3 \ldots i_{\mathcal{N}}\}$ and $\{j_2, j_3 \ldots j_{\mathcal{N}}\}$, resulting in the following state-space approximation for the inner product

$$\sum_{i_1=j_1=1}^{N_1} a_{\bar{i}} a_{\bar{j}} \propto \sum_{\alpha_1=\beta_1=1}^{L_1} \left\langle s_{\bar{\alpha}} \prod_{\mathcal{D}=2}^{\mathcal{N}} r_{\mathcal{D},i_{\mathcal{D}},\alpha_{\mathcal{D}}} \right\rangle_{\alpha_{\mathcal{D}}} \left\langle s_{\bar{\beta}} \prod_{\mathcal{D}=2}^{\mathcal{N}} r_{\mathcal{D},j_{\mathcal{D}},\beta_{\mathcal{D}}} \right\rangle_{\beta_{\mathcal{D}}} + \epsilon_{ij}, \tag{12}$$

where ϵ_{ij} denotes the approximation error.

The approximation (12) is more efficient than (11) as a result of omitting the numerous projections from state space to decoded weight-vectors in the estimation of the inner product. Furthermore, the simulation experiments presented below show that the variance of the inner-product approximation error increases when replacing the expectation value operations in (12) by an explicit inner product in state space, but otherwise an explicit evaluation of the inner product is possible in principle. This is expected because the averaging operations reduce the influence of state-space noise, $\epsilon_{\bar{\alpha}}$ in (6), on the approximate inner product. The computational cost of the expectation values in (12) is low thanks to the sparsity of index vectors, and the constant of proportionality depends on constant parameters only ($\mathcal{N}$, $L_{\mathcal{D}}$ and $\chi_{\mathcal{D}}$). Note that the expressions resulting from a different choice of constant indices, which represent the relationship to be compared, can be obtained by change of notation in the equations presented above. For example, instead of summing over $\alpha_1 = \beta_1$ in (12), it is possible to sum over $\alpha_2 = \beta_2$ and average over other indices in state space.

The generalised inner product approximation (12) and the encoding (3) and decoding (5) methods are available in the software implementation of generalised RI [47]. Next we present numerical results which demonstrate that the generalised methods that are introduced above are reasonable.

3 Simulation experiments

We study the generalised RI method presented above with numerical experiments. Ideally, analytical bounds should be derived for the error terms in (9) and the related approximation (12). However, the analysis is complicated because of the sparse ternary index vectors and the dependence on the structure of the data. Partial results are presented in “Appendix”, which may be useful for further development. The simulation experiments are conducted to verify that the proposed generalisation is feasible, and the results also demonstrate some characteristics of the method.

The approximation errors introduced when encoding and decoding array elements depend on some parameters, in particular the dimensionality of the array and the input data; the length of the index vectors, $L_{\mathcal{D}}$; the number of nonzero trits in the index vectors, $\chi_{\mathcal{D}}$; the dimension reduction, $\Pi_{\mathcal{D}} N_{\mathcal{D}} : \Pi_{\mathcal{D}} L_{\mathcal{D}}$; and the characteristics of the data that is encoded.

3.1 Verification and comparison with PCA

The first experiment is carried out to verify that the generalised methodology presented in Sect. 2 is feasible and furthermore to investigate how it compares to the well-known PCA algorithm on a basic semantic analysis task. We consider a sparse 5000×5050 band matrix, which is partially illustrated in the upper panel of Fig. 1.

The matrix has a diagonal band that is 50 elements wide. Therefore, nearby rows are similar (semantically related) vectors with high inner products compared to the inner products of more distant rows. A band matrix is used to simplify the graphical presentation and interpretation of the structure of the data and the reconstruction. However, because of the random projections involved in RI the particular structure of the data is not important. Similar RI results are expected for other data structures of comparable sparsity.

The middle panel of Fig. 1 illustrates the reconstructed matrix when 101 principal components are used, corresponding to a dimension reduction of about 25:1. This approximate representation of the band matrix is similar to the original, but the band on the main diagonal is more wide and additional band-like structures are visible. The PCA is performed with

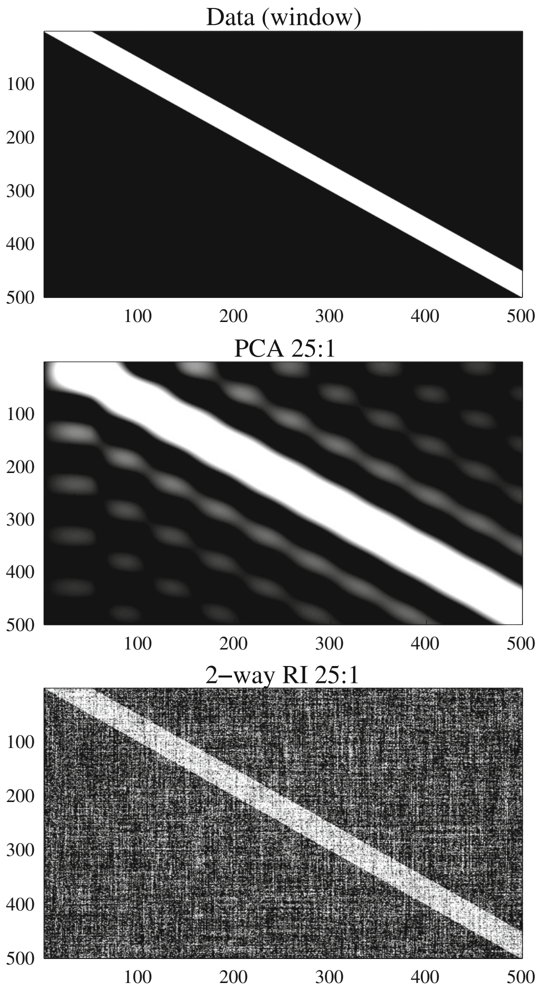


Fig. 1 Dimension reduction of a 5000×5050 band matrix (*top*) and reconstruction with PCA (*middle*) and two-dimensional RI (*bottom*). The dimension reduction is about 25:1 in both cases, which is obtained using 101 principal components, $\chi_{\mathcal{D}} = 8$ and $L_{\mathcal{D}} = 964$. In these three panels, a 500×500 window of the band matrix is displayed

MATLAB with double precision floating point numbers. The lower panel of Fig. 1 displays the matrix reconstructed using two-dimensional RI, $\mathcal{N} = 2$, for $\chi_{\mathcal{D}} = 8$ and $L_{\mathcal{D}} = 964$, which also corresponds to a dimension reduction of about 25:1. The RI analysis is based on signed 16-bit states, which in practice means that the RI representation of the matrix is about four times smaller than the PCA representation. The RI approximation of the matrix is similar to the original band matrix in the sense that the structure of the band is preserved, but there are significant approximation errors also in this case, which appears like noise in the figure.

The characteristics of the approximation errors introduced by PCA and RI are different. The effect of the error term in (9) is evident in the lower panel of Fig. 1 in the form of noise, which is an expected consequence of the random projections and distributional representation. Therefore, it is interesting to investigate the effect of the approximation errors on the semantic similarity of different rows. We calculate the average inner product between the 5000 different rows versus the distance between the rows; see Fig. 2.

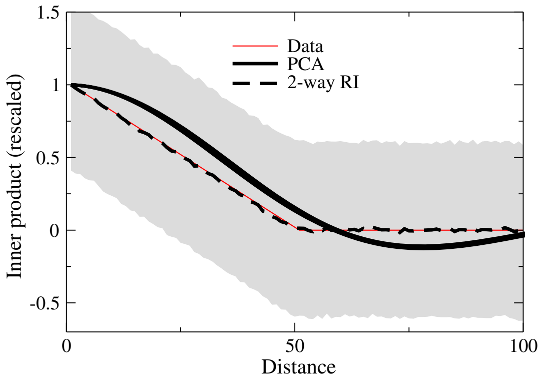


Fig. 2 Average inner product of rows versus the distance between the rows of the original band matrix (data), the PCA-approximated matrix (PCA) and the RI-approximated matrix (two-way RI). The *vertical axis* is normalised with the maximum average inner product in all three cases. *Shaded areas* denote ± 1 standard deviation of the PCA- and RI-approximated inner products

In the case of PCA, the inner products are calculated from the full reconstructed band matrix displayed in the middle panel of Fig. 1, and both the average and standard deviation (shaded area) of the inner product are displayed in Fig. 2. The inner products approximated with RI at a comparable dimension reduction are calculated with (12), which means that the inner products are calculated directly in state space, without reference to the reconstruction displayed in Fig. 1.

For comparison purposes, we calculate the inner products also from the reconstructed band matrix displayed in the lower panel of Fig. 1 and find that the average inner products and standard deviation are consistent with those displayed in Fig. 2. Furthermore, when omitting the state-space averaging operations in (12) we find that the standard deviation increases by a factor of more than two (data not shown), which confirms that the averaging operations reduce the influence of noise. These results motivate the approximation presented in (12), which reduces the computational cost and variance of RI-approximated inner products. These results are also in line with previous results showing that random projection preserves similarities of structures in the data [5,20].

The error introduced by the RI approximation has a significantly higher standard deviation than the error introduced by PCA. However, PCA introduces a variable bias in the average inner product, which is not observed in the RI results. When increasing the size and sparseness of the band matrix, we find that the standard deviation of RI-approximated inner products decreases and that the bias of the average inner product increases in the case of PCA (data not shown).

3.2 Decoding error and comparison with ordinary RI

The approximation errors of generalised RI and ordinary RI of distributional vectors are expected to be different because higher-order approximations involve additional random projections, each contributing to the noise of the representation. Therefore, we compare generalised and ordinary RI with simulation experiments. We consider an experiment where a matrix is approximated using ordinary and generalised RI at different sparsity levels. Matrices can be represented using ordinary, one-way RI if each column or row is treated as a vector, which is the method used in natural language processing.

We select a generic approach where each column of the matrix represents a *class*, and each row represents a possible *feature* of the classes. Therefore, the columns are feature vectors that can be used to calculate the similarity of the classes, and the columns in state space are distributional vectors that encode the similarity of the classes. This interpretation

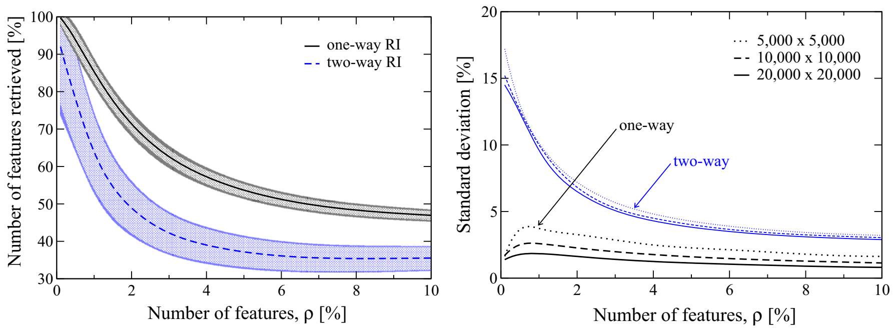


Fig. 3 Number of correctly decoded features approximated using one-way and two-way RI. The *vertical axis* of the *panel* on the *left-hand* (*right-hand*) side represents the average (standard deviation) of the relative number of correctly decoded features. In the case of the one-way (two-way) RI method, the higher standard deviation corresponds to a 5000 × 5000 matrix encoded in 1250 × 5000 (2500 × 2500) states, whereas the lower standard deviation corresponds to a 10000 × 10000 matrix encoded in 2500 × 10000 (5000 × 5000) states. The results presented in the panel on the *right-hand* side also correspond to these dimensionalities, and it includes an additional result for a 20000 × 20000 matrix that is approximated at a comparable dimension reduction of 4:1. Note that the higher dimensionality of the index vectors in the 20000 × 20000 case results in a lower standard deviation compared to the other two cases

and terminology is introduced to simplify the presentation. An integer sampled from the flat distribution [0, 10] is added to each element of the matrix, which simulates noise in the data that makes the matrix non-sparse. The non-sparse noise is introduced to make the experiment more challenging, and the choice of distribution is arbitrary since we have no particular application in mind. In addition to the noise, a relatively sparse set of high-value weights, $w_{ij} = 100$, are added to the matrix. The high-value weights simulate features of the classes, which we want to distinguish from the noise.

The number of features is selected to be proportional to the size of the matrix, $N_{\mathcal{D}}$, and we define the constant of proportionality as ρ. We vary the relative number of features, ρ, from 0.1 to 10% of the size of the matrix, $N_{\mathcal{D}}$. The array elements are decoded with (5) for each class, and the set of $\rho N_{\mathcal{D}}$ array elements with the highest values are identified. If not all elements representing encoded features of that class are identified in that set, some features are not correctly identified after decoding. In the following we present the number of features that are correctly identified. Unless stated otherwise, we use $\chi_{\mathcal{D}} = 8$ in the simulation experiments.

We find that the average number of correctly decoded features is practically *independent* of dimensionality, provided that the dimensionality is reasonably high ($\sim 10^3$ or higher because the variance explodes at low dimensionality). However, the *standard deviation* of the number of correctly decoded features decreases with increasing dimensionality. Therefore, if the dimension reduction, $\Pi_{\mathcal{D}} N_{\mathcal{D}} : \Pi_{\mathcal{D}} L_{\mathcal{D}}$, is kept constant and the number of encoded features is proportional to the size of the matrix, the effect of increasing the size of the matrix, and therefore the dimensionality of index vectors, is a reduction in the uncertainty of the number of correctly decoded features. This scaling behaviour is illustrated numerically in Fig. 3.

The average of the relative number of correctly decoded features is practically independent of the size of the matrix and the dimensionality of the index vectors, but the corresponding standard deviation decreases with increasing dimensionality of the index vectors. Note that the relative number of correctly decoded features first decreases with an increasing number of encoded features, as expected, and that it increases slightly for $\gtrsim 8\%$ features in the case

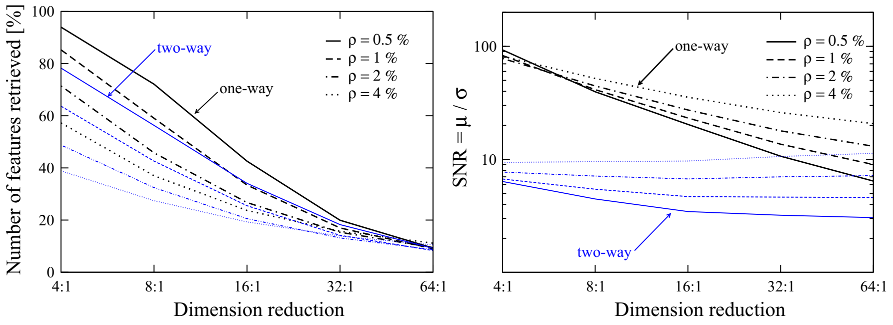


Fig. 4 Effect of the dimension reduction, $\Pi_{\mathcal{D}} N_{\mathcal{D}} : \Pi_{\mathcal{D}} L_{\mathcal{D}}$, on the relative number of correctly decoded features. The *panel* on the *left-hand side* presents the average relative number of correctly decoded features. The *panel* on the *right-hand side* shows the signal-to-noise ratio, which is defined as the average relative number of correctly decoded features, μ, divided by the corresponding standard deviation, σ. The size of the matrix, $N_{\mathcal{D}}$, is taken to be 64,000 × 64,000 for both the one-way and two-way RI method. At the maximum dimension reduction of 64:1, the matrix is encoded in 1000 × 64,000 (8,000 × 8,000) states using the one-way (two-way) RI method

of two-way RI. This effect is caused by the increasing probability of correctly identifying features by chance when the relative number of features increases. In the case of the ordinary one-way RI method, the standard deviation has a maximum at approximately 0.7–0.9% features.

3.2.1 Effect of dimension reduction

Next we modify the experiment outlined in Sect. 3.2 and investigate how a varying dimension reduction affects the possibility to correctly decode features of the classes. We vary the dimension reduction, $\Pi_{\mathcal{D}} N_{\mathcal{D}} : \Pi_{\mathcal{D}} L_{\mathcal{D}}$, from 4:1 to 64:1. The size of the matrix, $N_{\mathcal{D}}$, is kept constant, which implies that the number of features that are encoded in the classes is constant. The result of this simulation experiment is presented in Fig. 4. At increasing dimension reduction ratio the approximation accuracy decreases more quickly for the ordinary one-way RI method than the two-way generalisation. Therefore, the approximation error of two-way RI approaches that of one-way RI at high-dimension reduction ratios.

3.2.2 Effect of sparseness of the index vectors

Next, we modify the experiment outlined in Sect. 3.2 and investigate how the feature-decoding results presented in Fig. 3 depend on the number of nonzero trits, $\chi_{\mathcal{D}}$, in the index vectors. The parameter $\chi_{\mathcal{D}}$ governs the sparsity of the distributional representations, which affects both the computational cost of the encoding algorithm and the approximation error. In the results presented above we use $\chi_{\mathcal{D}} = 8$, which means that the index vectors have four positive and four negative trits. Figure 5 illustrates how the average number of correctly decoded features varies for different values of $\chi_{\mathcal{D}}$ and the relative number of encoded features, ρ. The choice $\chi_{\mathcal{D}} = 8$ is a good compromise because the accuracy increases insignificantly at higher values of $\chi_{\mathcal{D}}$ and the computational cost of the encoding and averaging operations is proportional to $\Pi_{\mathcal{D}} \chi_{\mathcal{D}}$. For example, the number of states associated with one matrix element is $\chi_1 \times \chi_2$ in the case of two-way RI, which implies that there is a quadratic dependence of

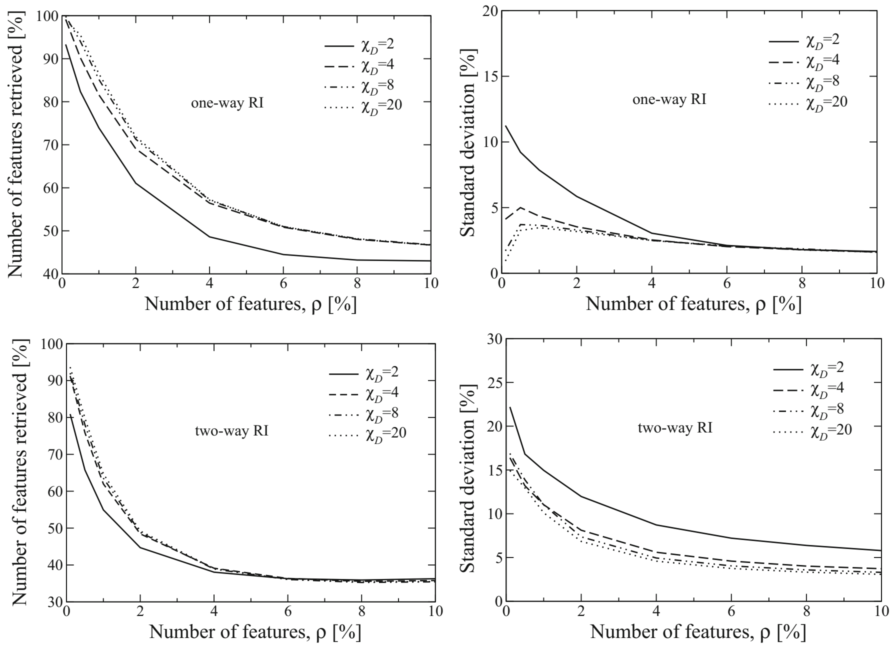


Fig. 5 Average number of correctly decoded features and the corresponding standard deviation for different numbers of nonzero trits in the index vectors, $\chi_{\mathcal{D}}$, and different relative number of encoded features, $\rho \in \{0.1,\ 0.5,\ 1,\ 2,\ 4,\ 6,\ 8,\ 10\}\%$. The matrix has a size of 5000×5000, and it is encoded using one-way and two-way RI at a dimension reduction of 4:1. The results for $\chi_{\mathcal{D}} = 8$ are identical to those presented in Fig. 3. The optimal choice for the sparseness of the index vectors is $\chi_{\mathcal{D}} \sim 8$ because the accuracy increases insignificantly for higher values, but the computational cost increases significantly for higher values of $\chi_{\mathcal{D}}$; see (4)

the computational cost on the number of nonzero trits in the index vectors, $\chi_{\mathcal{D}}$. This is the motivation for using $\chi_{\mathcal{D}} = 8$ in the simulations presented above.

3.3 Natural language processing example

Next, we apply the generalised RI method to a basic natural language processing task. In statistical models of natural language, it is common to construct a large co-occurrence matrix [52]. For example, this method is used in the Hyperspace Analogue to Language (HAL) [36,37] and in Latent Semantic Analysis (LSA) [35], which are two pioneering models in the field. In practical applications, the number of words can be hundreds of thousands. The number of documents or contexts is also high, otherwise the statistical basis will be insufficient for the analysis. Therefore, word co-occurrence matrices tend to be large objects. The simple example considered below uses more than 5 billion matrix elements that represent word–context relationships. Fortunately, co-occurrence matrices can be approximated to make the semantic analysis less computationally costly. It was first demonstrated by Kanerva, Kristoferson and Holst [31] that one-way RI can be used to effectively encode co-occurrence matrices for semantic analysis; see Sahlgren [42,43] and [30] for further details and developments.

The definition of "context" is model specific, but it typically involves a set of neighbouring words or one document. In HAL, the context is defined by a number of words that immediately surround a given word, whereas in LSA, the context is defined as the document where the word exists. Linguistically, the former relation can be described as a paradigmatic (semantic) relation, whereas the latter can be characterised as an associative (topical) relation. In the traditional RI algorithm, each word type that appears in the data is associated with a distributional vector, and each context is associated with a ternary index vector; see (1). If the context is defined in terms of the neighbouring words of a given word, which is the method that we use here, the distributional vectors are created by adding the index vectors (which can be weighted differently) of the nearest preceding and succeeding words every time a word occurs in the data [32]. If the context is defined as the document where the word exists, the distributional vectors are created by adding the index vectors of all of the documents where a word occurs, weighted with the frequency of the word in each document. In either case, a distributional vector is the sum of the weighted index vectors of all contexts where that word occurs.

The RI algorithm has been evaluated using various types of vocabulary tests, such as the synonymy part of the "Test of English as a Foreign Language" (TOEFL) [31,43]. In the following, we reconsider the synonym identification task presented by Kanerva, Kristoferson and Holst [31] with three changes. First, we want to compare the one-way and two-way RI methods. Therefore, we encode the co-occurrence matrix using both one-way and two-way RI. Second, while Kanerva, Kristoferson and Holst [31] used the LSA definition of context, we use a strategy similar to that in HAL and define the context as a window that spans ± 2 words away from the word itself. This method implies that for each occurrence of a word, there will be four additional word–word relationships encoded in the co-occurrence matrix. This strategy avoids the potential difficulty of defining document boundaries in streaming text, and it captures semantic relations between words rather than topical relations. The length of the context window is a parameter that affects the quantitative results presented here, but it is not essential for our qualitative discussion. The third difference compared with the study by Kanerva, Kristoferson and Holst [31] is that we do not introduce cut-offs on term frequencies to further improve the result. Words such as "the", "at" and "be" have high frequencies that render the occurrences of more interesting combinations less significant. We note that this effect is stronger for the two-way RI method than for the one-way RI method. We include the complete word–context spectrum, including the high-frequency relationships, and we present results for two different transformations of the spectrum. In one case, we directly encode the unaltered frequencies, and in the other case, we encode the square root of the frequencies. The square root decreases the relative significance of high frequencies, which improves the result and illustrates the importance of the definition of the weight in the feature extraction method.

We construct the co-occurrence matrix from 37,620 short high-school level articles in the TASA (Touchstone Applied Science Associates, Inc.) corpus. The text has been morphologically normalised so that each word appears in its base form [32]. The text contains 74,183 word types that are encoded in a co-occurrence matrix with one-way and two-way RI. In the case of one-way RI, we use index vectors of length 1000, so that the dimension reduction is $74{,}183 \times 74{,}183 \ : \ 1000 \times 74{,}183 \rightarrow 74 \ : \ 1$. In the case of two-way RI, we use a state array of size $1000 \times 74{,}183$, thereby maintaining the same dimension reduction ratio. We repeat the two-way RI calculations using a square state array of size 8612×8612. There are numerous misspellings (low-frequency words) in the corpus, and the most frequent word is "the", which occurs nearly 740,000 times. At the second place is "be" with just over 420,000 occurrences. Therefore, we define 32-bit states in the implementation of RI.

The task consists of eighty TOEFL synonym tests, which contains five words each. One example of a synonym test considered here is presented in Table 2. One out of the five words in each synonym test is given, and the task is to identify the synonym of that word among the

Table 2 Example of a TOEFL synonym test

Word	Number of occurrences
Essential (given)	855
Basic	1920
Ordinary	837
Eager	480
Possible	3348

The first word is given, and the task is to determine which of the four remaining words that is a synonym of that word. The number of occurrences of each word in the TASA (Touchstone Applied Science Associates, Inc.) corpus is also illustrated

other four words. There is only one correct synonym in each case, and consequently three incorrect alternatives.

The task to identify the correct synonym is addressed using the RI-encoded co-occurrence matrices, and the vector semantic comparison method described in Sect. 2.4. We consider 80 synonym tests, each comprising five words. Using ordinary RI, 38 out of the 80 synonym tests are solved correctly, meaning that the cosine of angle between the given word and correct synonym is maximum. Repeating the experiment with the square root of frequencies and ordinary RI, 43 out of the 80 synonym tests are solved correctly. Using two-way RI and a square state array of size 8612×8612 only 24 out of 80 synonym tests are solved correctly. Repeating the experiment with the square root of frequencies and two-way RI we obtain a similar result, 24 out of the 80 synonym tests are solved correctly. However, repeating the two-way RI experiment with the square root of frequencies and a state array of size $1000 \times 74{,}183$ we obtain 34 correct results out of 80.

These results can be further improved using other preprocessing methods, for example, by introducing weighted context windows and cut-offs on the encoded relationship frequencies [32], or by defining the weights as the logarithm of frequencies divided by the conditional entropy of the context given the word [35]. Furthermore, in order to enable numerical simulations on a standard PC we only consider higher-order RI of one distributed representation, while there is one distributed representation for each class/term in the case of one-way RI. This limitation can be avoided in large-scale applications of RI at data centres, possibly leading to more favourable results. One benefit of the two-way RI method is that words can be defined on the fly with a minimum impact on the storage space needed. This property is interesting for the analysis of streaming data, where many occasional features may exist in the data that are not relevant for the long-term analysis.

4 Conclusions

Random indexing is a form of random projection with particularly low computational complexity, thanks to the high sparsity of the index vectors and the straightforward distributed coding of information. RI has numerous applications and has proven useful for solving challenging problems without introducing much complexity.

Here we generalise ordinary RI (1) of distributional vectors [31,42] to RI of distributional arrays of arbitrary order, and we present results of simulation experiments with one- and two-way RI. The software implementation [47] of the key equations (3), (5) and (12) supports

N-way RI. Ordinary RI is used in numerous applications in natural language processing, where the possibility to approximate data in a compressed representation that can be updated incrementally at low computational cost and complexity in an online manner is useful. Furthermore, the compressed data structure can be used for semantic analysis by approximating inner products of distributional arrays at low computational cost using (12), without prior decoding of data. These properties make RI interesting for the analysis of streaming data, in particular when explicit storage of data is infeasible. Incremental processing of streaming data is not explicitly investigated in the simulation experiments presented in this work, but the data sets considered are encoded in an incremental fashion, one item after the other. The low computational complexity and incremental character of the encoding algorithm (3) make applications to streaming data straightforward. Furthermore, the possibility to extend the length of the random indices and therefore dynamically extend the number of properties that can be compressed in the state array makes RI interesting for analysis of streaming data. The generalisation of RI from distributional vectors to distributional arrays opens up for analysis of higher-order relationships, for example context- or time-dependent associations of terms in streaming text (third order), or the context- and time-dependent associations between terms (fourth order).

Our simulation results confirm the expectation that the approximation error is lower for ordinary one-way RI compared to two-way RI at constant size of the distributed representation. This is expected because each random index of an array is associated with additional random projections, which adds to the state-space noise. The benefit of two-way RI is that multiple classes of features can be encoded in a distributed representation of constant size and that new features can be defined on the fly with low impact on the storage space required. This property is interesting for the analysis of higher-order relationships between features derived from streaming data, where the number of potential features and relationships can be astronomical.

Higher-order RI can be applied to multiple distributional arrays, just like ordinary RI (1) typically is applied to a set of distributional vectors, $\mathbf{s}_i$. For example, in ordinary RI of co-occurrence matrices each column of the co-occurrence matrix is represented in a distributional vector, and each row of the co-occurrence matrix refers to an index vector that defines a random projection on the distributional vectors. This way the effect of state-space noise on the accuracy of term representations is minimised because each term is associated with a unique distributional vector. However, in this case the size of the distributed representation is proportional to the number of terms, which limits the scalability of the approach. Depending on the application requirements this principle can also be applied to higher-order RI when balancing between reasonable approximation errors, data storage demands and computational cost.

From a technical point of view we note that RI requires index vectors of high dimensionality (typically $n > 10^3$), otherwise the variance related to the approximate random projections explodes and renders the approach practically useless. This tendency is well described by Kanerva [30] in his paper about computation in "hyperdimensional" spaces, and we have observed a similar effect in earlier work on distributional models based on high-dimensional random projections [15,16]. For high-dimensional representations we find that the variances of approximated inner products and decoded weights decrease with increasing dimensionality and that the expectation values are practically invariant with respect to dimensionality. Furthermore, we find that the number of nonzero trits in the index vectors, $\chi_{\mathcal{D}}$, has an effect on the accuracy of RI. The accuracy increases notably when increasing $\chi_{\mathcal{D}}$ from two to four, but not much beyond $\chi_{\mathcal{D}} = 8$. Therefore, our simulation experiments suggest that $\chi_{\mathcal{D}} = 8$ is the preferred choice for this hyperparameter since the computational cost of the encoding and semantic analysis algorithms increase with $\chi_{\mathcal{D}}$.

In summary, RI is an incremental dimension reduction method that is computationally lightweight and well suited for online processing of streaming data not feasible to analyse with other, more accurate and complex methods [50]. For example, standard co-occurrence matrices in natural language processing applications can be extended with temporal information [26], linguistic relations [3,53] and structural information in distributed representations [9,59]. There have been few attempts at extending traditional matrix-based natural language processing methods to higher-order arrays due to the high computational cost involved. This is something that multidimensional RI is likely to facilitate.

Acknowledgements The TASA and TOEFL items have been kindly provided by Professor Thomas Landauer, University of Colorado. We thank Pentti Kanerva for comments on an early draft of the manuscript [48] and anonymous referees for useful comments and suggestions that helped us improve the clarity of the manuscript. The possibility to extend RI to two dimensions is mentioned in [30] at p. 153. This work is partially funded by the Kempe Foundations.

Appendix: Indifference property of high-dimensional ternary vectors

The tendency of random vectors in high-dimensional spaces to be *indifferent*, meaning that by chance they are unrelated ("nearly orthogonal"), is well known in the context of binary vectors [29,30]. Here we generalise that concept to ternary vectors $\{-1, 0, 1\}^n$, which are required for RI. These results were obtained in an attempt to derive bounds for the decoding error in (9) and are presented here for future use.

Consider the binary space $\{0, 1\}^n$ of vectors with length n, which have equal probability for the 0 and 1 states in each element. The distance, d, between two binary vectors can be defined as the number of nonzero bits in the bit-wise exclusive or (XOR) of the vectors. This distance is equivalent to the square of the Euclidean distance and it corresponds to the number of bits that are different in the two vectors, which is known as the Hamming distance. The number of vectors in the space that are at a distance d from a specific vector is given by the binomial coefficient

$$C(n, d) = \binom{n}{d},$$

because this is the number of different ways to choose (flip) d bits out of n. Therefore, the number of vectors at a certain distance from a reference vector follows the binomial distribution with a probability of $p = 1/2$, which has a mean of $n/2$ and a variance of $n/4$.

At high values of n the binomial distribution can be approximated using a normal distribution. If a distribution is approximately normal, the proportion within z standard deviations of the mean is $\mathrm{erf}(z/\sqrt{2})$. This relationship implies that the distance distribution is highly concentrated around the mean because the error function quickly approaches unity with increasing z. For example, 99.7% of the distances are within three standard deviations from the mean distance. Only one billionth (10^{-9}) of the distances deviate more than six standard deviations from the mean. The mean distance is $n/2$, and the standard deviation of the distance is $\sqrt{n}/2$. This implies that the mean distance is $\sqrt{n}$ standard deviations; for example, the mean distance is 31.6 standard deviations when $n = 1000$. A striking consequence of this distribution of distances is that practically all of the vectors in a high-dimensional binary space are located at distances that are approximately $n/2$ from any specific vector in the space.

In this work, we are interested in ternary vectors. Instead of bits that have two possible states, {0, 1}, we consider balanced trits that have three possible states $\{-1, 0, 1\}$. The introduction of a third state that has negative sign is crucial because it enables the sparse distributed coding of array elements in the states. This discussion concerns sparse ternary vectors of length n with k positive (1) and k negative (-1) trits, where $k \ll n/2$. Note that the index k in Sect. 2 is different from the symbol k defined here. The ternary space can be visualised as a subset of an inner product space where orthogonality is defined by a vanishing dot product between two vectors. With this definition of orthogonality it follows that an n-dimensional ternary space has at most n mutually orthogonal vectors. However, in a high-dimensional space, there are many more vectors that are indifferent. This result is analogous to the high probability of indifference between vectors in high-dimensional binary space. The total number, N, of ternary vectors of length n that has k positive and k negative elements is

$$N = \binom{n}{2k}\binom{2k}{k} = \binom{n}{k}\binom{n-k}{k}, \tag{13}$$

because there are $C(n, 2k)$ different ways to choose $2k$ nonzero trits and $C(2k, k)$ different ways to distribute the signs to the nonzero trits. The alternative (second) definition above can be interpreted in a similar way; there are $C(n, k)$ different ways to choose the positive trits and $C(n-k, k)$ ways to choose the negative trits or vice versa. How many of these N vectors are indifferent? The number of vectors that have an absolute value of the dot product, $d = |\langle\cdot,\cdot\rangle|$, with respect to any reference vector is (see proof below)

$$\begin{aligned} N(n,k,d) \simeq & \binom{n-2k}{2k-d}\binom{2k-d}{k}\binom{k}{k-d} \\ & \times\ {}_3F_2(-d,-k,-k;\ 1+k-d, 1+k-d;\ -1), \quad d \le k,\ n \gg k, \end{aligned} \tag{14}$$

where ${}_3F_2$ is a generalised hypergeometric function [40]. In the analysis leading to this expression we assume that $0 \le d \le k$ because we are only interested in indifferent vectors, and we assume that the vectors are sparse so that $n \gg k$. If we divide the number of vectors, $N(n,k,d)$, which has a specific value of d with respect to any reference vector, by the total number of vectors in the space, N, the result is the relative size of the space as a function of d. The relative size of the space is equivalent to the probability of randomly choosing a vector from the space that has a dot product of $\pm d$ with respect to a reference vector

$$P(n,k;\ \langle\cdot,\cdot\rangle = \pm d) \simeq N^{-1} N(n,k,d), \quad d \le k,\ n \gg k. \tag{15}$$

This result describes the probability for the randomly chosen vectors to be indifferent. The numbers N and $N(n,k,d)$ are extremely large (n is a high number). Therefore, for practical purposes, we make a series expansion of factors involving n in the limit $n \to \infty$. The result is

$$P(n,k;\ \langle\cdot,\cdot\rangle = \pm d) \simeq \frac{T}{n^d} \sum_{i=0}^{d} \frac{(k!)^4}{[(k-d+i)!]^2\,[(k-i)!]^2\,(d-i)!\,i!}, \tag{16}$$

$$\begin{aligned} T = {} & 1 - \frac{8k^2 + d^2 + d - 8kd}{2n} \\ & + \frac{1}{n^2}\left[2(1-2k)^2k^2 + \frac{d^4}{8} + \left(\frac{5}{12} - 2k\right)d^3 + \left(10k^2 - 4k + \frac{3}{8}\right)d^2 \right. \\ & \left. + \left(-16k^3 + 10k^2 - 2k + \frac{1}{12}\right)d\right] + \mathcal{O}(n^{-3}), \end{aligned} \tag{17}$$

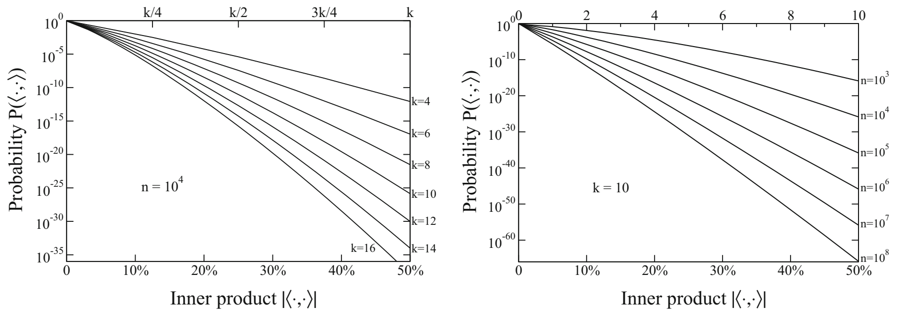


Fig. 6 Indifference property of high-dimensional ternary vectors $\{-1, 0, 1\}^n$. The *panel* on the *left-hand side* shows the probability (15) for inner products of sparse ternary vectors with length $n = 10^4$ and different numbers of nonzero trits, $2k$. The *panel* on the *right-hand side* shows the probability (15) for $k = 10$ and different lengths of the ternary vectors, n. In both cases the horizontal scale is normalised to the maximum value of the inner product, which is $2k$. Probabilities for absolute values of $\langle \cdot, \cdot \rangle$ greater than 50% of the maximum are excluded, because (16) is valid only for $d \leq k$. For $n = 10^4$ and $k = 4$, which are the ternary vectors with a length of ten thousand elements with four positive and four negative trits, the probability that a randomly generated vector has an inner product of four with respect to a reference vector is approximately 10^{-12}. The probability of an inner product of minus four is also approximately 10^{-12}. Similarly, for $n = 10^4$ and $k = 12$ the probability of 50% overlap ($\langle \cdot, \cdot \rangle = \pm 12$) is approximately 10^{-30}

where the terms in T originate from the series expansion. The assumptions $d \leq k$ and $n \gg k$ are to be respected in applications of this result, which has not been presented elsewhere as far as we know.

The following example illustrates the indifference property of high-dimensional ternary vectors. Let $n = 10^4$ and $k = 10$, which are typical parameters used [30,31]. It follows from (15) that 96% of the space is orthogonal with respect to any reference vector, and less than 4% of the space has a dot product of $+1$ or -1 (see Table 3). Only 7×10^{-9} of the space has a dot product with a magnitude greater than or equal to four, which corresponds to approximately 20% nonzero trits in common. With 25% common trits ($d = 5$ and $k = 10$), the relative size of the space is 2×10^{-11}. Therefore, most of the space is approximately orthogonal to any particular vector in the space. Analogously, the dot products of vectors that are randomly sampled from the space are given by the probability (15). The probabilities for $n = 10^4$ and some different values of k are illustrated in Fig. 6.

Proof

Here we derive the relation used in (14). The total number, N, of ternary vectors of length n that has k positive and k negative trits is

$$N = \binom{n}{2k}\binom{2k}{k}, \tag{18}$$

because there are $C(n, 2k)$ different ways to choose $2k$ nonzero trits and $C(2k, k)$ different ways to distribute the signs to the nonzero trits. How many of these N vectors have a dot product that is nearly zero; i.e. how many of them are indifferent? Let $d = |\langle \cdot, \cdot \rangle|$ be the absolute value of the dot product between two vectors. For simplicity we restrict the analysis to $0 \leq d \leq k$, because we are interested in indifferent vectors only. This restriction does not affect the accuracy of the result. We assume also that the vectors are sparse so that $n \gg k$. Imagine a fixed reference vector that is picked at random from the space of N vectors. This reference vector has k positive trits, k negative trits and $n - 2k$ trits that are zero. The large

Table 3 Indifference of vectors in the high-dimensional space $\{-1, 0, 1\}^n$

$2k$	$\langle\cdot,\cdot\rangle$	$n = 10^2$		$n = 10^3$		$n = 10^4$	
		P_{sim}	P	P_{sim}	P	P_{sim}	P
4	0	8.5e−1	8.47e−1	9.8e−1	9.84e−1	∼1.0	9.98e−1
	±1	7.3e−2	7.29e−2	7.9e−3	7.93e−3	8.0e−4	7.99e−4
	±2	2.0e−3	1.94e−3	2.0e−5	1.99e−5	2.0e−7	2.00e−7
8	0	5.5e−1	*5.17e−1	9.4e−1	9.38e−1	9.9e−1	9.94e−1
	±1	1.9e−1	1.90e−1	3.0e−2	3.05e−2	3.2e−3	3.20e−3
	±2	2.8e−2	2.74e−2	3.9e−4	3.86e−4	4.0e−6	3.99e−6
	±3	2.0e−3	1.96e−3	2.4e−6	2.44e−6	2.5e−9	2.49e−9
	±4	7.4e−5	7.42e−5	8.2e−9	8.22e−9	$<10^{-10}$	8.3e−13
12	0	3.5e−1	–	8.7e−1	8.65e−1	9.9e−1	9.86e−1
	±1	2.3e−1	–	6.4e−2	6.37e−2	7.1e−3	7.10e−3
	±2	7.9e−2	–	2.0e−3	1.99e−3	2.2e−5	2.17e−5
	±3	1.6e−2	–	3.4e−5	3.44e−5	3.7e−8	3.69e−8
	±4	2.0e−3	–	3.6e−7	3.64e−7	$<10^{-10}$	3.8e−11
16	0	2.5e−1	–	7.8e−1	*7.73e−1	9.7e−1	9.75e−1
	±1	2.0e−1	–	1.0e−1	1.02e−1	1.3e−2	1.25e−2
	±2	1.1e−1	–	5.9e−3	5.94e−3	7.1e−5	7.09e−5
	±3	4.3e−2	–	2.0e−4	2.01e−4	2.3e−7	2.34e−7
	±4	1.1e−2	–	4.4e−6	4.44e−6	4.9e−10	5.0e−10
20	0	2.0e−1	–	6.9e−1	*6.72e−1	9.6e−1	9.61e−1
	±1	1.8e−1	–	1.4e−1	1.39e−1	1.9e−2	1.93e−2
	±2	1.2e−1	–	1.3e−2	1.31e−2	1.8e−4	1.75e−4
	±3	6.5e−2	–	7.4e−4	7.36e−4	9.6e−7	9.55e−7
	±4	2.7e−2	–	2.8e−5	2.78e−5	3.5e−9	3.49e−9

Tabulated here is the probability, P, in (16) for different values of the vector length, n, and number of nonzero elements, $2k$. These probabilities are to be compared with the corresponding probabilities obtained from explicit numerical simulations, P_{sim}. Entries marked with an asterisk demonstrate the effect of neglecting contributions to the inner product arising from higher-order trit combinations (like $\langle\cdot,\cdot\rangle = \cdots + 1 \times 1 \cdots - 1 \times 1 \cdots + 1 \times 1 \cdots = 1$) in the analysis leading to (16). The series expansion is marginally applicable in the case $n = 10^2$ for low values of k, and $n \gg k$ is violated for high k

majority of vectors with $\langle\cdot,\cdot\rangle = \pm d$ with respect to this reference vector will have d trits that coincides with the $2k$ nonzero trits of the reference vector, and the remaining $2k-d$ nonzero trits will be distributed among the $n-2k$ trits that are zero in the reference vector. There are additional vectors with the same value of d, because cancellations of type $1+1-1 = 1$ result from higher-order coincidences. The relative number of such vectors is insignificant, and we therefore neglect them here. This simplification is justified with a numerical calculation that is presented below. The selection of $2k-d$ nonzero trits out of $n-2k$ gives a factor of $C(n-2k, 2k-d)$. Then remains the question how many possibilities there are to select those $2k-d$ nonzero trits from the $2k$ nonzero trits in the reference vector, and how many combinations that arise because of signs. These questions are not independent, because the number of ways to choose $2k-d$ trits from $2k$ trits depends on the number of $+1$ trits that are chosen, and the relative number of $+1$ trits that are chosen will affect also the number of possible permutations. Accounting for these constraints the number of vectors is

$$N(n,k,d) \simeq \binom{n-2k}{2k-d} \sum_{n_+=k-d}^{k} \binom{k}{n_+}\binom{k}{2k-d-n_+}\binom{2k-d}{n_+}, \quad d \leq k,\ n \gg k, \tag{19}$$

where n_+ denotes the number of positive trits that are chosen from the $2k$ nonzero trits in the reference vector. The number of negative trits chosen is $n_- = 2k - d - n_+$. The sum in (19) arises because there are multiple choices for the number of positive trits to choose from the reference vector. At most k positive trits can be chosen, i.e. all positive trits. The lower limit of $n_+ = k - d$ corresponds to the maximum value for the number of negative trits chosen, $n_- = k$. The first factor in the sum, $C(k, n_+)$, accounts for the number of ways to choose n_+ positive trits from the k positive trits in the reference vector. Similarly, the second factor accounts for the number of ways to choose n_- negative trits from the k negative trits in the reference vector. The last factor accounts for sign permutations when distributing the chosen trits to the $2k - d$ nonzero trits that are selected by the prefactor. If we divide the number of vectors, $N(n, k, d)$, that have a specific value of d with respect to any reference vector, with the total number of vectors in the space, N, the result is the relative size of the space as a function of d. The relative size of the space is equivalent to the probability of randomly choosing a vector from the space that has a dot product of $\pm d$ with respect to the reference vector. Since the number of positive and negative signs are fixed, the combinatorial problem solved here has a hypergeometric character. The sum in (19) can be replaced with a generalised hypergeometric function. The result of that substitution is presented in (15).

Numerical results for the dot product between a reference vector and 10^{12} randomly chosen ternary vectors are presented in Table 3.

These numerical results confirm the analytical result. Observe, however, that the accuracy of the analytical result is poor for low values of n and high values of k, as indicated in the table. This is connected to the assumption that $n \gg k$ in the analysis above.

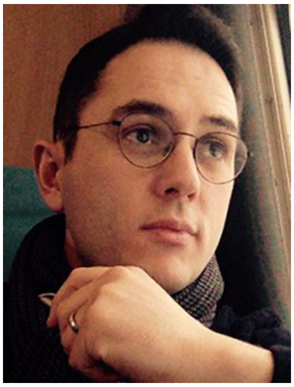
References

1. Achlioptas D (2003) Database-friendly random projections: Johnson–Lindenstrauss with binary coins. J Comput Syst Sci 66(4):671–687
2. Baraniuk RG (2011) More is less: signal processing and the data deluge. Science 331(6018):717–719
3. Baroni M, Lenci A (2010) Distributional memory: a general framework for corpus-based semantics. Comput Linguist 36(4):673–721
4. Berry M, Mezher D, Sameh A, Philippe B (2003) Parallel computation of the singular value decomposition. In: Kontoghiorghes EJ (ed) Handbook on parallel computing and statistics. Chapman and Hall/CRC, Boca Raton, pp 117–164
5. Bingham E, Mannila H (2001) Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining, KDD '01, ACM, pp 245–250
6. Boyd MWM (2010) Randomized algorithms for matrices and data. Found Trends Mach Learn 3(2):123–224
7. Broder AZ (1997) On the resemblance and containment of documents. In: Compression and complexity of sequences (SEQUENCES'97), IEEE Computer Society, pp 21–29
8. Bullinaria J, Levy J (2012) Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. Behav Res Methods 44(3):890–907. doi:10.3758/s13428-011-0183-8
9. Clark S, Pulman S (2007) Combining symbolic and distributional models of meaning. In: Proceedings of the AAAI spring symposium on quantum interaction, pp 52–55
10. Cohen T (2008) Exploring MEDLINE space with random indexing and pathfinder networks. AMIA Ann Symp Proc 2008:126–130
11. Cohen T, Widdows D (2009) Empirical distributional semantics: methods and biomedical applications. J Biomed Inform 42(2):390–405

12. Damljanovic D, Petrak J, Cunningham H (2010) Random indexing for searching large rdf graphs. In: The proceedings of the 7th extended semantic web conference (ESWC 2010), Springer Verlag, Heraklion, Greece
13. Dasgupta S, Gupta A (2003) An elementary proof of a theorem of Johnson and Lindenstrauss. Random Struct Algorithms 22(1):60–65
14. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
15. Emruli B, Gayler R, Sandin F (2013) Analogical mapping and inference with binary spatter codes and sparse distributed memory. In: The 2013 international joint conference on neural networks (IJCNN), pp 1–8
16. Emruli B, Sandin F (2014) Analogical mapping with sparse distributed memory: a simple model that learns to generalize from examples. Cogn Comput 6(1):74–88
17. Fradkin D, Madigan D (2003) Experiments with random projections for machine learning. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, KDD ’03, ACM, pp 517–522
18. Frankl P, Maehara H (1988) The Johnson–Lindenstrauss lemma and the sphericity of some graphs. J Comb Theory Ser B 44(3):355–362
19. Fronza I, Sillitti A, Succi G, Terho M, Vlasenko J (2013) Failure prediction based on log files using random indexing and support vector machines. J Syst Softw 86(1):2–11
20. Goel N, Bebis G, Nefian A (2005) Face recognition experiments with random projection. Biom Technol Human Identif II 5779:426–437
21. Gorman J, Curran JR (2006) Scaling distributional similarity to large corpora. In: Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics, pp 361–368
22. Halko N, Martinsson PG, Tropp JA (2011) Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev 53(2):217–288
23. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, second edition: data mining, inference, and prediction, Springer Series in Statistics, 2nd ed. 2009. Springer, New york
24. Henriksson A, Hassel M, Kvist M (2011) Diagnosis code assignment support using random indexing of patient records—a qualitative feasibility study. In: Peleg M, Lavrač N, Combi C (eds) Artificial intelligence in medicine, number 6747 in lecture notes in computer science. Springer, Berlin Heidelberg, pp 348–352
25. Johnson W, Lindenstrauss J (1984) Extensions of Lipschitz maps into a Hilbert space. Contemp Math 26:189–206
26. Jurgens D, Stevens K (2009) Event detection in blogs using temporal random indexing. In: Proceedings of the workshop on events in emerging text types, eETTs ’09, Association for Computational Linguistics, Morristown, NJ, USA, pp 9–16
27. Jurgens D, Stevens K (2010) The s-space package: an open source package for word space models. In: Proceedings of the ACL 2010 system demonstrations, ACLDemos ’10, Association for Computational Linguistics, Morristown, NJ, USA, pp 30–35
28. Kane DM, Nelson J (2014) Sparser Johnson–Lindenstrauss transforms. J ACM 61(1):4–23
29. Kanerva P (1988) Sparse distributed memory. MIT Press, Cambridge
30. Kanerva P (2009) Hyperdimensional computing: an introduction to computing in distributed representation with High-Dimensional random vectors. Cogn Computation 1(2):139–159
31. Kanerva P, Kristoferson J, Holst A (2000) Random indexing of text samples for latent semantic analysis. In: Proceedings of the 22nd annual conference of the cognitive science society, p 1036
32. Karlgren J, Sahlgren M (2001) From words to understanding. Foundations of real-world intelligence. CSLI Publications, Stanford
33. Kaski S (1998) Dimensionality reduction by random mapping: fast similarity computation for clustering. In: The 1998 IEEE international joint conference on neural networks proceedings, 1998. IEEE world congress on computational intelligence, vol 1, pp 413–418
34. Kolda TG, Bader BW (2009) Tensor decompositions and applications. SIAM Rev 51(3):455–500
35. Landauer TK, Dumais ST (1997) A solution to plato’s problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychol Rev 104(2):211–240
36. Lund K, Burgess C (1996) Producing high-dimensional semantic spaces from lexical co-occurrence. Beh Res Methods 28(2):203–208
37. Lund K, Burgess C, Atchley RA (1995) Semantic and associative priming in a high-dimensional semantic space. In: Cognitive science proceedings, pp 660–665
38. Matoušek J (2008) On variants of the Johnson–Lindenstrauss lemma. Random Struct Algorithms 33(2):142–156

39. Mitliagkas I, Caramanis C, Jain P (2013) Memory limited, streaming pca. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in neural information processing systems, vol 26. Curran Associates, Inc., pp 2886–2894
40. Olver FW, Lozier DW, Boisvert RF, Clark CW (2010) NIST handbook of mathematical functions. Cambridge University Press, New York, NY, USA. See also the NIST digital library of mathematical functions http://dlmf.nist.gov
41. Papadimitriou CH, Tamaki H, Raghavan P, Vempala S (1998) Latent semantic indexing: a probabilistic analysis. In: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems, PODS '98, ACM, pp 159–168
42. Sahlgren M (2005) An introduction to random indexing. In: Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering
43. Sahlgren M (2006) The word-space model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces, doctoral thesis, Stockholm University
44. Sahlgren M (2008) The distributional hypothesis. Ital J Linguist 20(1):33–54
45. Sahlgren M, Holst A, Kanerva P (2008) Permutations as a means to encode order in word space. In: Proceedings of the 30th annual meeting of the cognitive science society
46. Sahlgren M, Karlgren J (2009) Terminology mining in social media. In: Proceedings of the 18th ACM conference on information and knowledge management, CIKM '09, ACM, New York, NY, USA, pp 405–414
47. Sandin F (2016) N-way random indexing implementation. doi:10.5281/zenodo.53766
48. Sandin F, Emruli B, Sahlgren M (2011) Incremental dimension reduction of tensors with random index, CoRR arXiv:1103.3585
49. Staff Science (2011) Challenges and opportunities. Science 331(6018):692–693
50. Sun J, Tao D, Papadimitriou S, Yu PS, Faloutsos C (2008) Incremental tensor analysis: Theory and applications. ACM Trans Knowl Discov Data 2(3):11. doi:10.1145/1409620.1409621
51. Turney PD (2002) Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In: Proceedings of the 40th annual meeting on Association for Computational Linguistics, ACL '02, Association for Computational Linguistics, Stroudsburg, PA, USA, pp 417–424
52. Turney PD, Pantel P (2010) From frequency to meaning : vector space models of semantics. J Artif Intell Res 37:141–188
53. Van de Cruys T (2009) A non-negative tensor factorization model for selectional preference induction. In: Proceedings of the workshop on geometrical models of natural language semantics, Association for Computational Linguistics, Athens, Greece, pp 83–90
54. Vasuki V, Cohen T (2010) Reflective random indexing for semi-automatic indexing of the biomedical literature. J Biomed Inform 43(5):694–700
55. Velldal E (2011) Predicting speculation: a simple disambiguation approach to hedge detection in biomedical literature. J Biomed Semant 2(S–5):S7
56. Vempala SS (2004) The random projection method. American Mathematical Society, Providence
57. Wan M, Jönsson A, Wang C, Li L, Yang Y (2012) Web user clustering and web prefetching using random indexing with weight functions. Knowl Inf Syst 33(1):89–115
58. Widdows D, Ferraro K (2008) Semantic vectors: a scalable open source package and online technology management application. In: Proceedings of the sixth international language resources and evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco
59. Yeung H, Tsang P (2004) Distributed representation of syntactic structure by tensor product representation and non-linear compression. In: Proceedings of the 19th national conference on artificial intelligence, AAAI'04, AAAI Press, pp 437–442

Fredrik Sandin is an associate professor in industrial electronics at LTU. He holds an M.Sc. in Engineering Physics and a Ph.D. in Physics. He received a "New-Talents Award" (2004) from the International School of Subnuclear Physics in Erice for "An original work in theoretical physics"; two post-doctoral scholarships in theoretical physics (2008–2009) and brain-inspired computing (2010–2011); and the Gunnar Öquist Fellowship (2014) from the Kempe Foundations. His research focuses on cognitive computing and neuromorphic engineering.



Blerim Emruli is a senior researcher at SICS, where his work focuses on machine learning and its application to solve real-world industrial problems. He received his M.Sc. degree (2009) with focus in artificial intelligence from Dalarna University, Sweden, and the Ph.D. degree (2014) with focus in cognitive computing from Luleå University of Technology, Sweden. His Ph.D. thesis "Ubiquitous Cognitive Computing: A Vector Symbolic Approach" extends the previous studies on computational models of cognition and meaning using vector-based representations

Magnus Sahlgren is a senior researcher at SICS and co-founder of Gavagai. He holds a Ph.D. in computational linguistics from Stockholm University and has worked on computational models of meaning since 2000. Sahlgren's dissertation "The Word-Space Model" was awarded the prize for the most prominent scholarly achievement of 2006 at the Stockholm University Faculty of Humanities. His current research is situated at the intersection between computational linguistics, machine learning and artificial intelligence.