Distributional Semantics

Magnus Sahlgren

Pavia, 12 September 2012

Recap

Words-by-regions matrices and LSA

Words-by-words-matrices and HAL

Dependency-based models

Recap

Words-by-regions matrices and LSA

Words-by-words-matrices and HAL

Dependency-based models

Recap

Words-by-regions matrices and LSA

Words-by-words-matrices and HAL

Dependency-based models

Recap

Words-by-regions matrices and LSA

Words-by-words-matrices and HAL

Dependency-based models

This lecture

Word space models in the lab

vs.

Word space models in the Real World
(dumbing down and scaling up)

This lecture

Word space models in the lab

vs.

Word space models in the Real World
(dumbing down and scaling up)

This lecture

Word space models in the lab

vs.

Word space models in the Real World
(dumbing down and scaling up)

Word space models in the lab

Corpora (e.g. TASA, ANC, BNC, WaC...)

- Small- to medium-sized
- Static
- Editorial (for the most part)

Word space models in the lab

Corpora (e.g. TASA, ANC, BNC, WaC...)

- Small- to medium-sized
- Static
- Editorial (for the most part)

Word space models in the lab

Corpora (e.g. TASA, ANC, BNC, WaC...)

- Small- to medium-sized
- Static
- Editorial (for the most part)

Word space models in the lab

Corpora (e.g. TASA, ANC, BNC, WaC...)

- Small- to medium-sized
- Static
- Editorial (for the most part)

Word space models in the lab

Processing cost is not critical

Processing dependencies are acceptable, and sometimes even preferred

Word space models in the lab

Processing cost is not critical

Processing dependencies are acceptable, and sometimes even preferred

Word space models in the lab

Koptjevskaja-Tamm & Sahlgren, 2012

	General (BNC)	*News (Reuters)*	*Blogg (Spinn3r)*
	boiling	warm	castoff
	distilled	inclement	bomsight
hot	brackish	wintry	warm
	drinking	changeable	scald
	cold	mild	bottled
	hot	inclement	cream
	franco-prussian	mild	cube
cold	boer	warm	rink
	iran-iraq	wintry	floe
	napoleonic	changeable	skating

Computational Semantics in the Real World

Data

- Big Data
- Dynamic (streaming) data
- Non-editorial (i.e. noisy)

Computational Semantics in the Real World

Data

- Big Data
- Dynamic (streaming) data
- Non-editorial (i.e. noisy)

Computational Semantics in the Real World

Data

- Big Data
- Dynamic (streaming) data
- Non-editorial (i.e. noisy)

Computational Semantics in the Real World

Data

- Big Data
- Dynamic (streaming) data
- Non-editorial (i.e. noisy)

Example 1

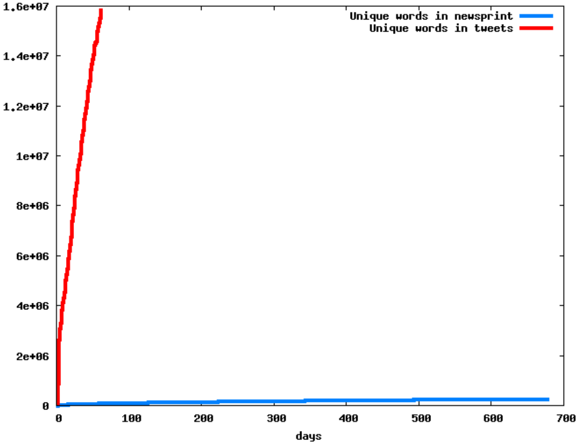
recommend	
recomend	0.972
reccomend	0.968
reccommend	0.941
looove	0.870
loooove	0.863
lurve	0.850
love	0.846
looooove	0.836

Example 2

good		bad	
great	0.91	weird	0.86
prefect	0.83	sucky	0.86
perfect	0.83	scary	0.86
pristine	0.81	cool	0.85
stable	0.80	nasty	0.84
grat	0.80	dumb	0.84
fantastic	0.80	sad	0.84
flawless	0.79	lame	0.84
mint	0.79	creepy	0.84
immaculate	0.79	stupid	0.84

Computational Semantics in the Real World

New words in New Text



Computational Semantics in the Real World

Processing cost is critical

Processing dependencies are a liability

Computational Semantics in the Real World

Processing cost is critical

Processing dependencies are a liability

Computational Semantics in the Real World

Problem: the huge co-occurrence matrix

Solution: dimension reduction

Computational Semantics in the Real World

Problem: the huge co-occurrence matrix

Solution: dimension reduction

Dimension Reduction

Matrix factorization: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), etc.

Random projection: $A'_{m,k} = A_{m,n} R_{n,k}$
where $k \ll n$

Simple statistics (e.g. column variance)

Dimension Reduction

Matrix factorization: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), etc.

Random projection: $A'_{m,k} = A_{m,n} R_{n,k}$
where $k \ll n$

Simple statistics (e.g. column variance)

Dimension Reduction

Matrix factorization: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA), Non-negative Matrix Factorization (NMF), etc.

Random projection: $A'_{m,k} = A_{m,n} R_{n,k}$
where $k \ll n$

Simple statistics (e.g. column variance)

Dimension Reduction

Problem: the huge co-occurrence matrix

Solution: don't build the huge co-occurrence matrix!

Dimension Reduction

Problem: the huge co-occurrence matrix

Solution: don't build the huge co-occurrence matrix!

Pre-defined contexts

One approach: use a small set of pre-defined contexts (words, tuples, etc.)

Another approach: Random Indexing

Pre-defined contexts

One approach: use a small set of pre-defined contexts (words, tuples, etc.)

Another approach: Random Indexing

Random Indexing

Designed to be on-line, scalable and efficient

Based on Pentti Kanerva's work on sparse distributed memory

Can be used with documents, words, tuples (and anything else) as contexts

Random Indexing

Designed to be on-line, scalable and efficient

Based on Pentti Kanerva's work on sparse distributed memory

Can be used with documents, words, tuples (and anything else) as contexts

Random Indexing

Designed to be on-line, scalable and efficient

Based on Pentti Kanerva's work on sparse distributed memory

Can be used with documents, words, tuples (and anything else) as contexts

Random Indexing

A standard co-occurrence matrix uses one unique dimension per context

$$
\begin{array}{c}
 \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m
\end{array}
\begin{array}{c}
\begin{array}{cccccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} \end{array} \\
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

Random Indexing

Random Indexing uses several *non-unique* dimensions per context

$$
\begin{array}{c c}
 & \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \end{array} \\
\begin{array}{c} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} &
\left[\begin{array}{ccccc}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0
\end{array}\right]
\end{array}
$$

$c_1 = [r_1, r_3]$

Random Indexing

Random Indexing uses several *non-unique* dimensions per context

$$
\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array}
\begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \end{array}
$$

$c_1 = [r_1, r_3] \qquad c_2 = [r_1, r_4]$

Random Indexing

Random Indexing uses several *non-unique* dimensions per context

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & r_4 & r_5 \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

$c_1 = [r_1, r_3]$ $\quad$ $c_2 = [r_1, r_4]$ $\quad$ $c_2 = [r_2, r_4]$

Random Indexing

The dimensionality is not defined by the number of contexts: it is a parameter (normally on the order of thousands)

This means that in Random Indexing, the dimensionality *never increases*

Random Indexing

The dimensionality is not defined by the number of contexts: it is a parameter (normally on the order of thousands)

This means that in Random Indexing, the dimensionality *never increases*

Random Indexing

Which dimensions a certain context is represented by is selected at random (hence the name random indexing)

The risk of randomly selecting the exact same dimensions for two different contexts is negligible *(we'll come back to why)*

Random Indexing

Which dimensions a certain context is represented by is selected at random (hence the name random indexing)

The risk of randomly selecting the exact same dimensions for two different contexts is negligible *(we'll come back to why)*

Random Indexing

The distributed random representation for a context is called a random index vector:

$$a_{ij} = \begin{cases} +1 & \text{with probability } \frac{\epsilon/2}{k} \\ 0 & \text{with probability } \frac{k-\epsilon}{k} \\ -1 & \text{with probability } \frac{\epsilon/2}{k} \end{cases}$$

where ϵ is the number of active dimensions and k is the dimensionality

Random Indexing

The distributed random representation for a context is called a random index vector:

$$a_{ij} = \begin{cases} +1 & \text{with probability } \frac{\epsilon/2}{k} \\ 0 & \text{with probability } \frac{k-\epsilon}{k} \\ -1 & \text{with probability } \frac{\epsilon/2}{k} \end{cases}$$

where ϵ is the number of active dimensions and k is the dimensionality

Random Indexing

Random index vectors are:

- High-dimensional
- Sparse (a small number of active dimensions)
- Ternary (active dimensions have either $+1$ or -1)
- Random

Random Indexing

Random index vectors are:

- High-dimensional
- Sparse (a small number of active dimensions)
- Ternary (active dimensions have either $+1$ or -1)
- Random

Random Indexing

Random index vectors are:

- High-dimensional
- Sparse (a small number of active dimensions)
- Ternary (active dimensions have either $+1$ or -1)
- Random

Random Indexing

Random index vectors are:

- High-dimensional
- Sparse (a small number of active dimensions)
- Ternary (active dimensions have either $+1$ or -1)
- Random

Random Indexing

Random index vectors are:

- High-dimensional
- Sparse (a small number of active dimensions)
- Ternary (active dimensions have either $+1$ or -1)
- Random

Random Indexing

Context vectors are accumulated *incrementally*

- Each word has a context vector (initially empty)
- Each context is assigned a random index vector
- Every time a word occurs, the context's random index vector is added to the word's context vector

Random Indexing

Context vectors are accumulated *incrementally*

- Each word has a context vector (initially empty)
- Each context is assigned a random index vector
- Every time a word occurs, the context's random index vector is added to the word's context vector

Random Indexing

Context vectors are accumulated *incrementally*

- Each word has a context vector (initially empty)
- Each context is assigned a random index vector
- Every time a word occurs, the context's random index vector is added to the word's context vector

Random Indexing

Context vectors are accumulated *incrementally*

- Each word has a context vector (initially empty)
- Each context is assigned a random index vector
- Every time a word occurs, the context's random index vector is added to the word's context vector

Random Indexing

Words-by-regions-style

Every time a word occurs, add the region's index vector to the word's context vector

Words-by-words-style

Every time a word occurs, add the index vectors of the surrounding words to the word's context vector

Random Indexing

Words-by-regions-style
Every time a word occurs, add the region's index vector to the word's context vector

Words-by-words-style
Every time a word occurs, add the index vectors of the surrounding words to the word's context vector

Random Indexing

Word w_1 occurs in document d_1 with index vector:
$[+1, 0, -1, ..., 0]$

$$
\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array}
\begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\
\left[\begin{array}{ccccc} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{array}\right] \end{array}
$$

Random Indexing

Word w_1 occurs in document d_1 with index vector:
$[+1, 0, -1, ..., 0]$

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{array}\right] \end{array}$$

Random Indexing

Word w_1 occurs in document d_1 with index vector:
$[+1, 0, -1, ..., 0]$

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\ \left[\begin{array}{ccccc} +1 & 0 & -1 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{array} \right] \end{array}$$

Random Indexing

Word w_1 occurs in the sequence $w_2 w_1 w_3$
where w_2 has index vector: $[+1, 0, -1, ..., 0]$
and w_3 has index vector: $[+1, -1, 0, ..., 0]$

$$
\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array}
\begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\
\left[\begin{array}{ccccc}
0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 0
\end{array}\right] \end{array}
$$

Random Indexing

Word w_1 occurs in the sequence	$w_2 w_1 w_3$
where w_2 has index vector:	$[+1, 0, -1, ..., 0]$
and w_3 has index vector:	$[+1, -1, 0, ..., 0]$

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} \end{array}$$

Random Indexing

Word w_1 occurs in the sequence $w_2 w_1 w_3$
where w_2 has index vector: $[+1, 0, -1, ..., 0]$
and w_3 has index vector: $[+1, -1, 0, ..., 0]$

$$\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array} \begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{array}\right] \end{array}$$

Random Indexing

Word w_1 occurs in the sequence	$w_2 w_1 w_3$
where w_2 has index vector:	$[+1, 0, -1, ..., 0]$
and w_3 has index vector:	$[+1, -1, 0, ..., 0]$

$$
\begin{array}{c} \\ w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_m \end{array}
\begin{array}{c} \begin{array}{ccccc} r_1 & r_2 & r_3 & \dots & r_k \end{array} \\
\left[\begin{array}{ccccc}
+2 & -1 & -1 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & 0
\end{array} \right] \end{array}
$$

Random Indexing

A context vector is a sum of random index vectors

Captures the same information as standard co-occurrence matrices

but using considerably less dimensions (= scalable)

Random Indexing

A context vector is a sum of random index vectors

Captures the same information as standard co-occurrence matrices

but using considerably less dimensions (= scalable)

Random Indexing

A context vector is a sum of random index vectors

Captures the same information as standard co-occurrence matrices

but using considerably less dimensions (= scalable)

Random Indexing

The underlying maths

The same as in random projection (the *Johnson-Lindenstrauss lemma*):

$$A'_{m,k} = A_{m,n} R_{n,k}$$

but without building the (huge) co-occurrence matrix $A_{m,n}$

Random Indexing

The underlying maths

The same as in random projection (the *Johnson-Lindenstrauss lemma*):

$$A'_{m,k} = A_{m,n} R_{n,k}$$

but without building the (huge) co-occurrence matrix $A_{m,n}$

Random Indexing

The underlying maths

The same as in random projection (the *Johnson-Lindenstrauss lemma*):

$$A'_{m,k} = A_{m,n} R_{n,k}$$

but without building the (huge) co-occurrence matrix $A_{m,n}$

Random Indexing

The underlying maths

In a standard co-occurrence matrix, the contexts are *orthogonal* to each other

The distributed random index vectors are *nearly* orthogonal to each other

Selecting dimensions at random in a high-dimensional space will *approximate* orthogonality

Random Indexing

The underlying maths

In a standard co-occurrence matrix, the contexts are *orthogonal* to each other

The distributed random index vectors are *nearly* orthogonal to each other

Selecting dimensions at random in a high-dimensional space will *approximate* orthogonality

Random Indexing

The underlying maths

In a standard co-occurrence matrix, the contexts are *orthogonal* to each other

The distributed random index vectors are *nearly* orthogonal to each other

Selecting dimensions at random in a high-dimensional space will *approximate* orthogonality

Random Indexing

The underlying maths

In a standard co-occurrence matrix, the contexts are *orthogonal* to each other

The distributed random index vectors are *nearly* orthogonal to each other

Selecting dimensions at random in a high-dimensional space will *approximate* orthogonality

Random Indexing

Lab:

- Build a Random Indexing model

Random Indexing

Lab:

- Build a Random Indexing model

Random Indexing

Additional refinements:

- Word order by permutations
- Syntagmatic relations by inverse permutations

Random Indexing

Additional refinements:

- Word order by permutations
- Syntagmatic relations by inverse permutations

Random Indexing

Additional refinements:

- Word order by permutations
- Syntagmatic relations by inverse permutations

Random Permutations

A general technique for encoding structure in distributed representations

Without sacrificing scalability, efficiency and performance

One application: encoding word order

Random Permutations

A general technique for encoding structure in distributed representations

Without sacrificing scalability, efficiency and performance

One application: encoding word order

Random Permutations

A general technique for encoding structure in distributed representations

Without sacrificing scalability, efficiency and performance

One application: encoding word order

Word Order

A context vector is a sum of (random) index vectors

Word order does not matter when summing vectors:

"the baby fed the bear"
$c(fed) = r(the) + r(baby) + r(the) + r(bear)$

"the bear ate the baby"
$c(ate) = r(the) + r(bear) + r(the) + r(baby)$

Word Order

A context vector is a sum of (random) index vectors

Word order does not matter when summing vectors:

"the baby fed the bear"
$c(fed) = r(the) + r(baby) + r(the) + r(bear)$

"the bear ate the baby"
$c(ate) = r(the) + r(bear) + r(the) + r(baby)$

Word Order

A context vector is a sum of (random) index vectors

Word order does not matter when summing vectors:

"the baby fed the bear"
$c(fed) = r(the) + r(baby) + r(the) + r(bear)$

"the bear ate the baby"
$c(ate) = r(the) + r(bear) + r(the) + r(baby)$

Word Order

A context vector is a sum of (random) index vectors

Word order does not matter when summing vectors:

“the baby fed the bear”
$c(\mathit{fed}) = r(\mathit{the}) + r(\mathit{baby}) + r(\mathit{the}) + r(\mathit{bear})$

“the bear ate the baby”
$c(\mathit{ate}) = r(\mathit{the}) + r(\mathit{bear}) + r(\mathit{the}) + r(\mathit{baby})$

Encoding word order

One solution: circular convolution (HRR, BEAGLE)

- Context vectors are accumulated from *n*-grams (up to 7)
- The *n*-grams are bound by circular convolution ($*$) and an auxiliary random vector (Φ)

$$c(fed) = (r(baby) * \Phi) + (r(the) * r(baby) * \Phi) + \ldots$$

Encoding word order

One solution: circular convolution (HRR, BEAGLE)

- Context vectors are accumulated from *n*-grams (up to 7)
- The *n*-grams are bound by circular convolution ($*$) and an auxiliary random vector (Φ)

$$c(fed) = (r(baby) * \ \Phi) + (r(the) * \ r(baby) * \ \Phi) + ...$$

Encoding word order

One solution: circular convolution (HRR, BEAGLE)

- Context vectors are accumulated from *n*-grams (up to 7)
- The *n*-grams are bound by circular convolution ($*$) and an auxiliary random vector (Φ)

$$c(fed) = (r(baby) * \Phi) + (r(the) * r(baby) * \Phi) + \ldots$$

Encoding word order

One solution: circular convolution (HRR, BEAGLE)

- Context vectors are accumulated from *n*-grams (up to 7)
- The *n*-grams are bound by circular convolution ($*$) and an auxiliary random vector (Φ)

$$c(fed) = (r(baby) * \ \Phi) + (r(the) * \ r(baby) * \ \Phi) + \ldots$$

Encoding word order

One solution: circular convolution (HRR, BEAGLE)

- Context vectors are accumulated from *n*-grams (up to 7)
- The *n*-grams are bound by circular convolution ($*$) and an auxiliary random vector (Φ)

$$c(fed) = (r(baby) * \Phi) + (r(the) * r(baby) * \Phi) + \ldots$$

Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Direction vectors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) *one* step if a word occurs to the left or right of the focus word (cf. HAL)

$$c(fed) = \Pi^{-1} r(the) + \Pi^{-1} r(baby) + \Pi\, r(the) + \Pi\, r(bear)$$

Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Direction vectors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) *one* step if a word occurs to the left or right of the focus word (cf. HAL)

$$c(fed) = \Pi^{-1} r(the) + \Pi^{-1}\, r(baby) + \Pi\, r(the) + \Pi\, r(bear)$$

Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Direction vectors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) *one* step if a word occurs to the left or right of the focus word (cf. HAL)

$$c(fed) = \Pi^{-1} r(the) + \Pi^{-1} r(baby) + \Pi\, r(the) + \Pi\, r(bear)$$

Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Direction vectors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) *one* step if a word occurs to the left or right of the focus word (cf. HAL)

$$c(\textit{fed}) = \Pi^{-1} r(\textit{the}) + \Pi^{-1}\, r(\textit{baby}) + \Pi\, r(\textit{the}) + \Pi\, r(\textit{bear})$$

Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Order vetors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) as many steps as the distance to the focus word

$$c(fed) = \Pi^{-2} r(the) + \Pi^{-1}\, r(baby) + \Pi\, r(the) + \Pi^{2}\, r(bear)$$

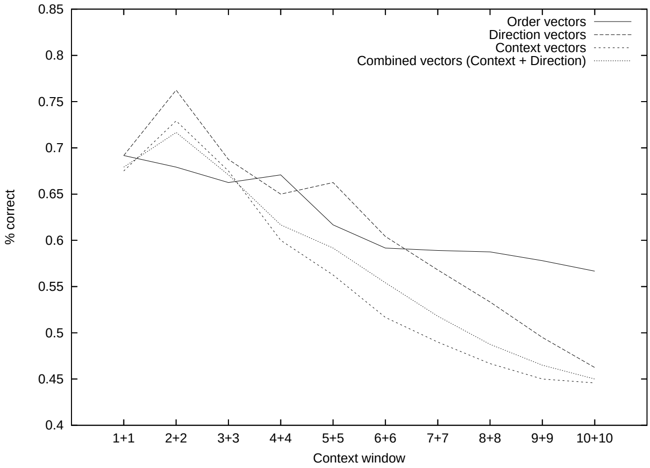
Encoding word order

Another solution: permutation of vector coordinates

- Permutation operation (Π): rotation of vector coordinates.
- Order vetors: rotate the random index vectors to the left (Π^{-n}) or to the right (Π^{n}) as many steps as the distance to the focus word

$$c(fed) = \Pi^{-2} r(the) + \Pi^{-1}\, r(baby) + \Pi\, r(the) + \Pi^{2}\, r(bear)$$

Context window size



Paired-associative memory

Given a trace vector $t = (x_1 \odot y_1) + (x_2 \odot y_2) + (x_3 \odot y_3) + \ldots$

and a probe vector x_i,

find the associate y_i from a set of possible random vectors.

Paired-associative memory

Given a trace vector $t = (x_1 \odot y_1) + (x_2 \odot y_2) + (x_3 \odot y_3) + \ldots$

and a probe vector x_i,

find the associate y_i from a set of possible random vectors.

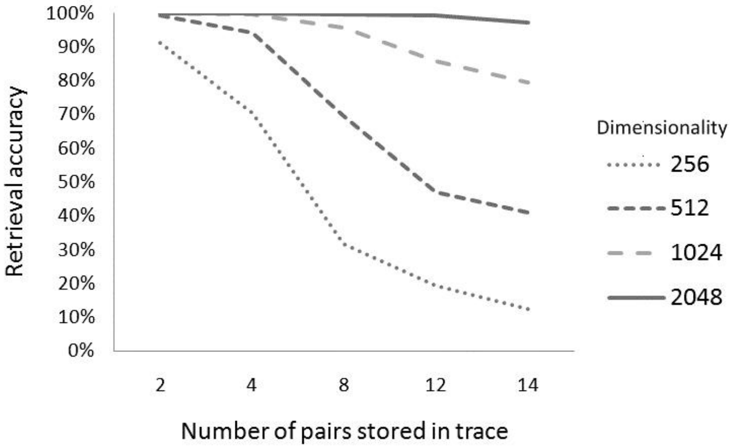
Paired-associative memory

Given a trace vector $t = (x_1 \odot y_1) + (x_2 \odot y_2) + (x_3 \odot y_3) + \ldots$

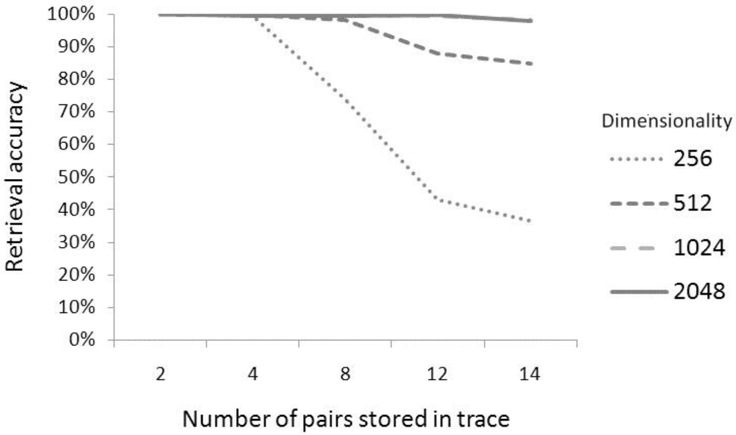
and a probe vector x_i,

find the associate y_i from a set of possible random vectors.

Circular convolution



Random permutations



Results

TOEFL synonym scores, averaged over 3 runs using different initializations of the random vectors.

BEAGLE score: 57.81%

Random permutations: $\approx 80\%$

(Roget's thesaurus: 78.75%, WordNet $< 25\%$
Jarmasz & Szpakowicz, 2003)

Results

TOEFL synonym scores, averaged over 3 runs using different initializations of the random vectors.

BEAGLE score: 57.81%

Random permutations: $\approx$ 80%

(Roget's thesaurus: 78.75%, WordNet $<$ 25%
Jarmasz & Szpakowicz, 2003)

Results

TOEFL synonym scores, averaged over 3 runs using different initializations of the random vectors.

BEAGLE score: 57.81%

Random permutations: $\approx$ 80%

(Roget's thesaurus: 78.75%, WordNet $<$ 25%
Jarmasz & Szpakowicz, 2003)

Results

TOEFL synonym scores, averaged over 3 runs using different initializations of the random vectors.

BEAGLE score: 57.81%

Random permutations: $\approx$ 80%

(Roget's thesaurus: 78.75%, WordNet $<$ 25%
Jarmasz & Szpakowicz, 2003)

Order and directional neighbors

Extract frequent left and right neighbors by using the inverse permutation.

Whenever "baby fed" occurs, $\Pi^{-1}\, r(\mathit{baby})$ is added to $c(\mathit{fed})$.

To retrieve "baby" from $c(\mathit{fed})$ we will compare $\Pi c(\mathit{fed})$ to all *random index vectors*.

Order and directional neighbors

Extract frequent left and right neighbors by using the inverse permutation.

Whenever "baby fed" occurs, $\Pi^{-1}\, r(baby)$ is added to $c(fed)$.

To retrieve "baby" from $c(fed)$ we will compare $\Pi c(fed)$ to all *random index vectors*.

Order and directional neighbors

Extract frequent left and right neighbors by using the inverse permutation.

Whenever "baby fed" occurs, $\Pi^{-1}\, r(baby)$ is added to $c(fed)$.

To retrieve "baby" from $c(fed)$ we will compare $\Pi c(fed)$ to all *random index vectors*.

Examples

KING

Word before	Word after
luther .24	queen .43
martin .22	england .25
become .17	midas .16
french .14	france .15
dr .13	jr .14

Examples

PRESIDENT

Word before	Word after
vice .69	roosevelt .22
become .23	johnson .20
elect .20	nixon .18
goodway .09	kennedy .15
former .09	lincoln .15

Random Permutations

Lab:

- Build a Random Permutations model

Random Permutations

Lab:

- Build a Random Permutations model

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Random Indexing

Summary

- Implicit (built-in) dimension reduction
- Any kind of context
- Permutations to handle structure
- Parallelizable
- Generalizable to tensors

Lab

Use GSDM and S-space to:

- Experiment with Random Indexing
- Experiment with Random Permutations

Lab

Use GSDM and S-space to:

- Experiment with Random Indexing
- Experiment with Random Permutations

Lab

Use GSDM and S-space to:

- Experiment with Random Indexing
- Experiment with Random Permutations