Random Positive-Only Projections
PPMI-Enabled Incremental Semantic Space Construction

Behrang QasemiZadeh and Laura Kallmeyer
DFG CRC 991, Heinrich-Heine Universität Düsseldorf

*SEM 2016, August 11-12, 2016

1

Outline

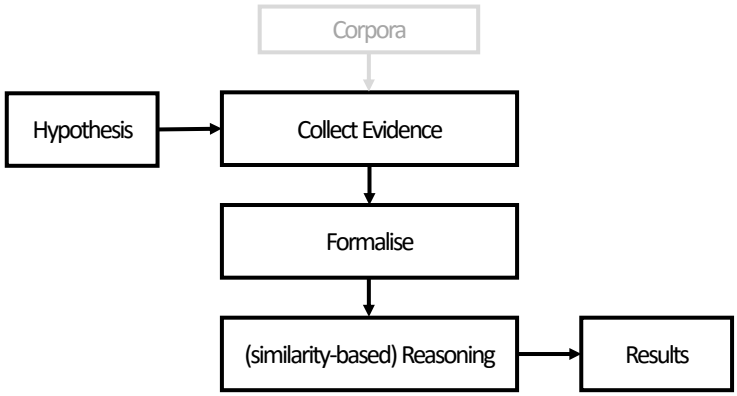
- Introduction + Motivation
- The Method
- Experiments and Results
  - Parameters of the Method
- Summary

2

Introduction and Motivation

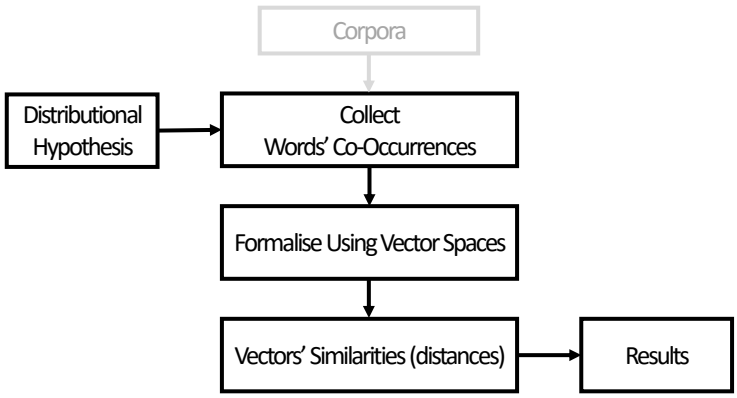
3

Data-driven,
Corpus-based,
Distributional NLP



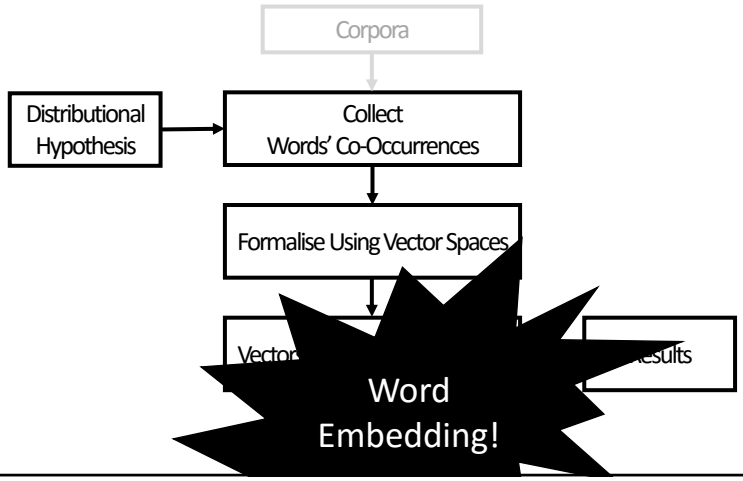
4

The Distributional Hypothesis



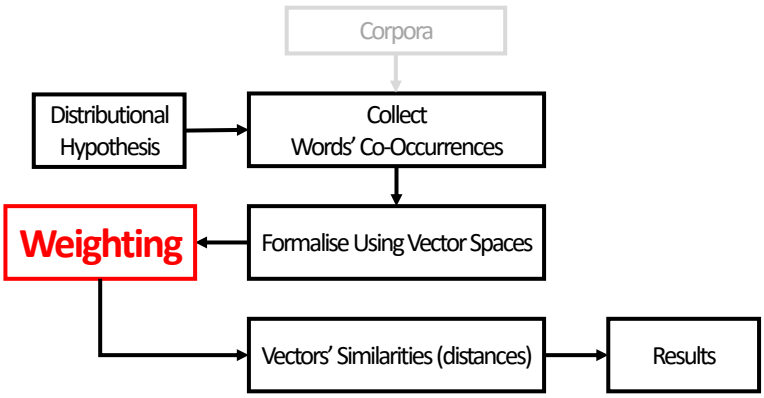
5

The Distributional Hypothesis
Word Space Models
Word Vectors …



6

The Weighting Process



7

The Weighting Process

- **Problem:** Raw frequencies are not that informative when it comes to meanings:
  - Not all the co-occurrences are important;
  - Different co-occurrences have different importance/impact on meanings.

- **Solution:** A *measure of association* is used to convert raw frequencies into more informative numbers.

8

The Weighting Process

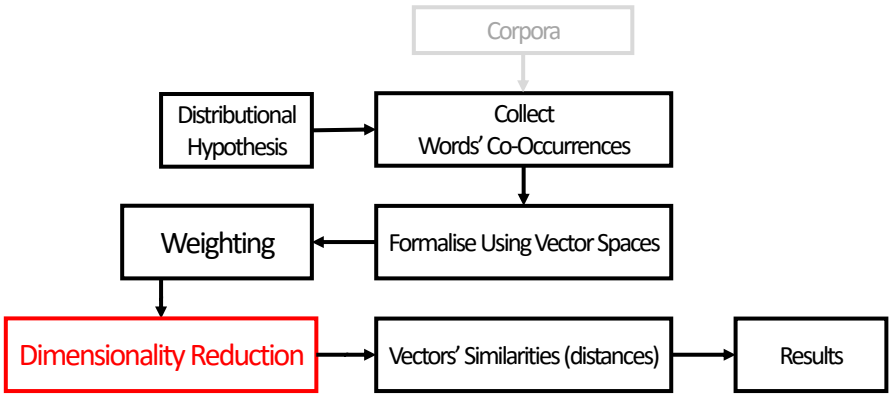
- Positive Pointwise Mutual Information (PPMI) is an effective weighting method in word spaces.
- PPMI is an estimate of how much more two words co-occur than it is expected by chance.
- For a component $\boldsymbol{v_{xy}}$ in an $\boldsymbol{n}$-dimensional vector space consisting of $\boldsymbol{p}$ vectors

$$PPMI(v_{xy}) = max\left(0, \log_2 \frac{P(x,y)}{P(x)P(y)}\right)$$

$$= \max\left(0, \log_2 \frac{v_{xy} \times \sum_{i=1}^{p} \sum_{j=1}^{n} v_{ij}}{\sum_{i=1}^{p} v_{iy} \times \sum_{i=1}^{n} v_{xi}}\right).$$

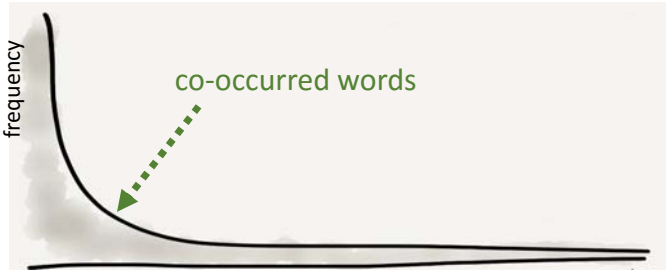
9

Dimensionality Reduction



10

Dimensionality Reduction

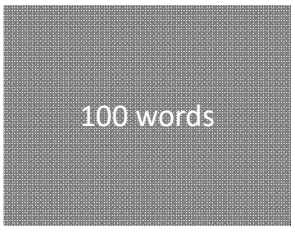


- Due to the *heavy-tailed* distribution of words, the number of context words escalates when new words are added to the model.
- Adding a new word requires adding additional dimensions to the vector space.

11

Dimensionality Reduction

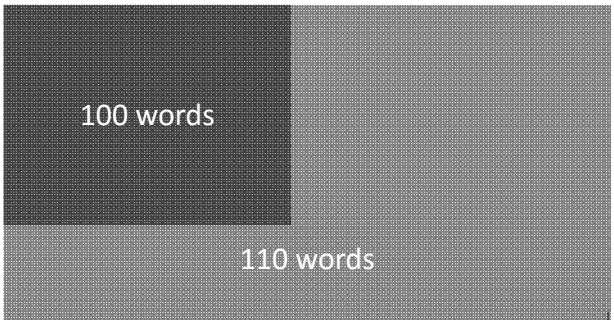
Dimensionality



12

Dimensionality Reduction

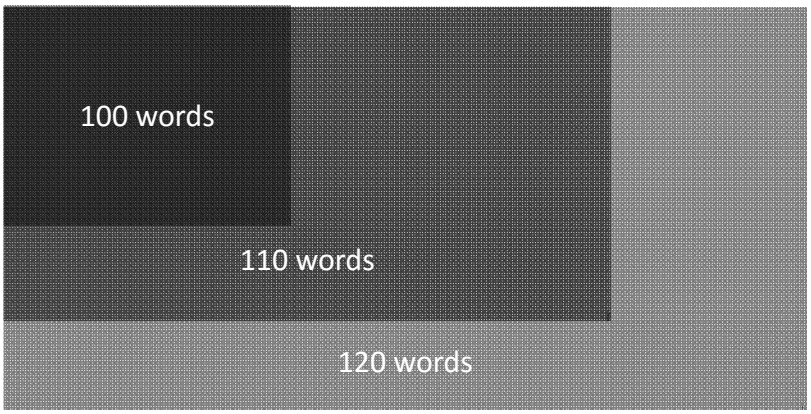
Dimensionality Increasing →



13

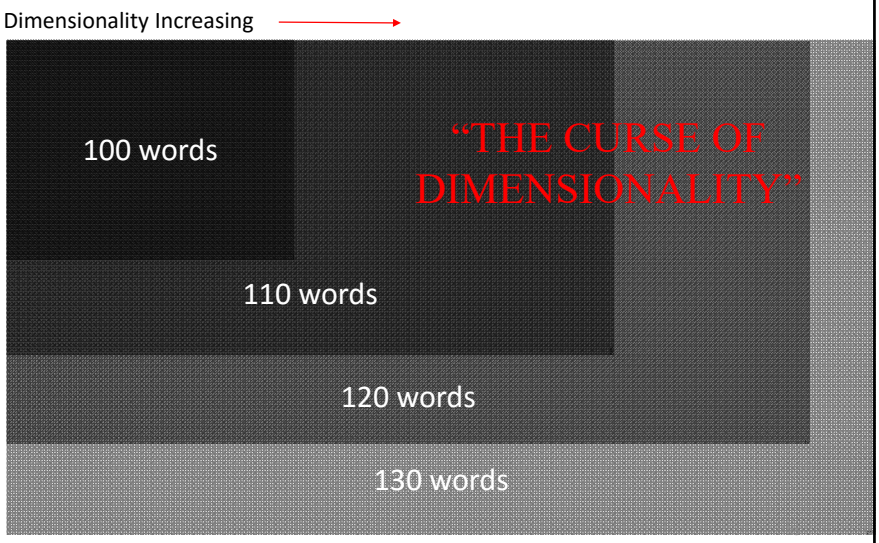
Dimensionality Reduction

Dimensionality Increasing →



14

Dimensionality Reduction

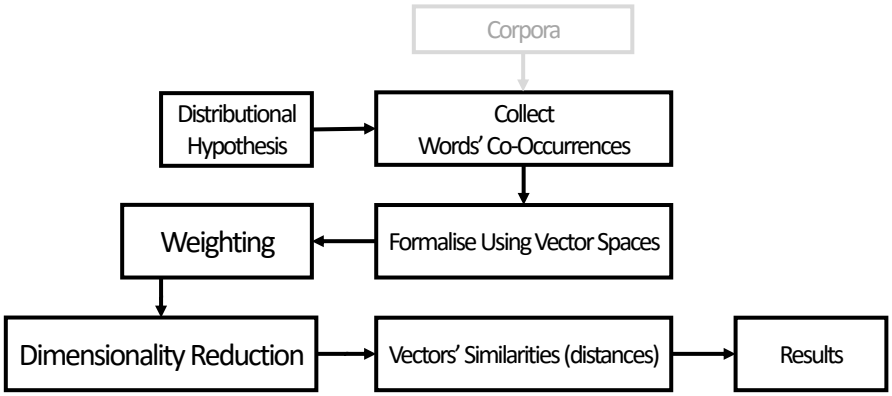


The Curse of Dimensionality

- The proportional impact of co-occurrences in vector similarities reduces when the dimension of a vector space model increases.
  - The meaning of two words is distinguishable only if they appear in completely different context.
- In real world applications, models become computationally intractable due to their huge size.

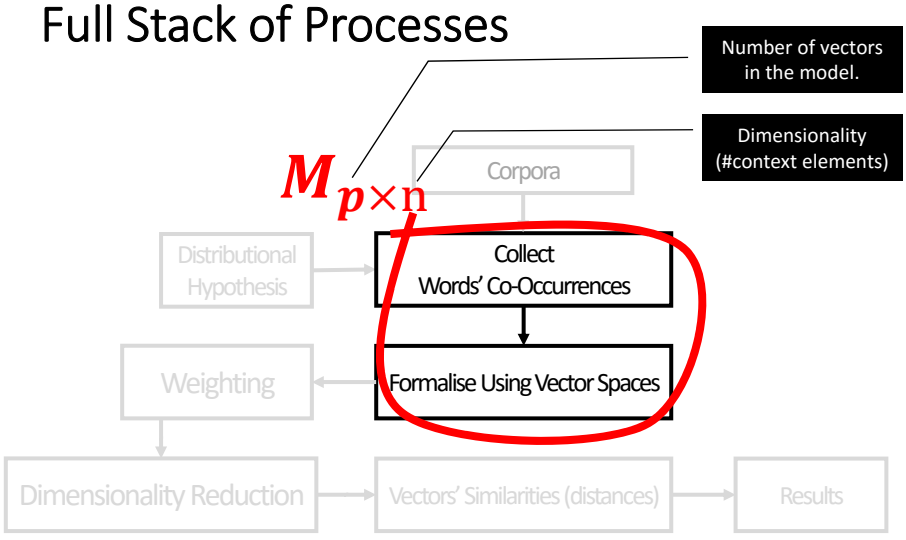
16

Full Stack of Processes



17

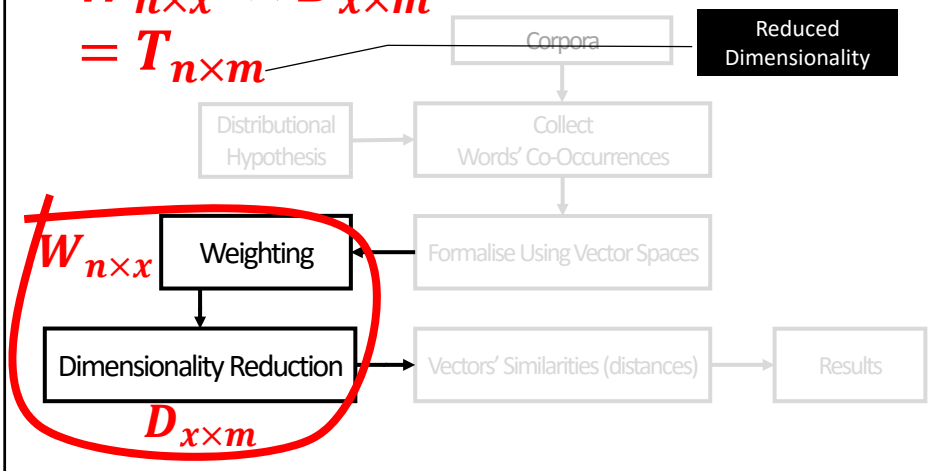
Full Stack of Processes



18

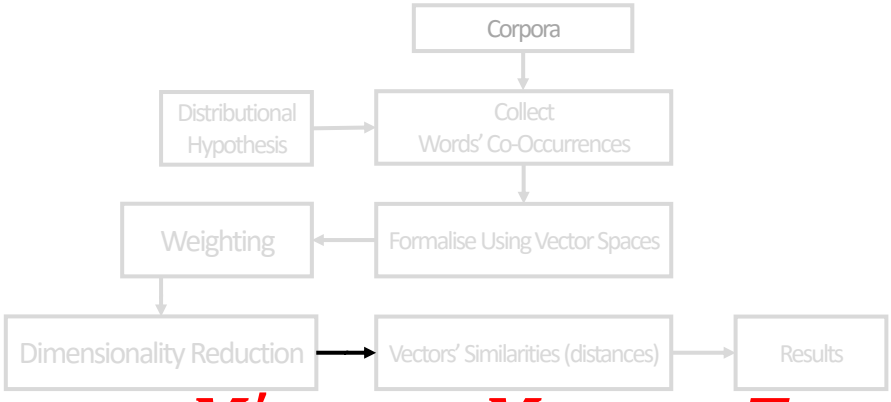
Full Stack of Processes

$$W_{n\times x} \times D_{x\times m} = T_{n\times m}$$



19

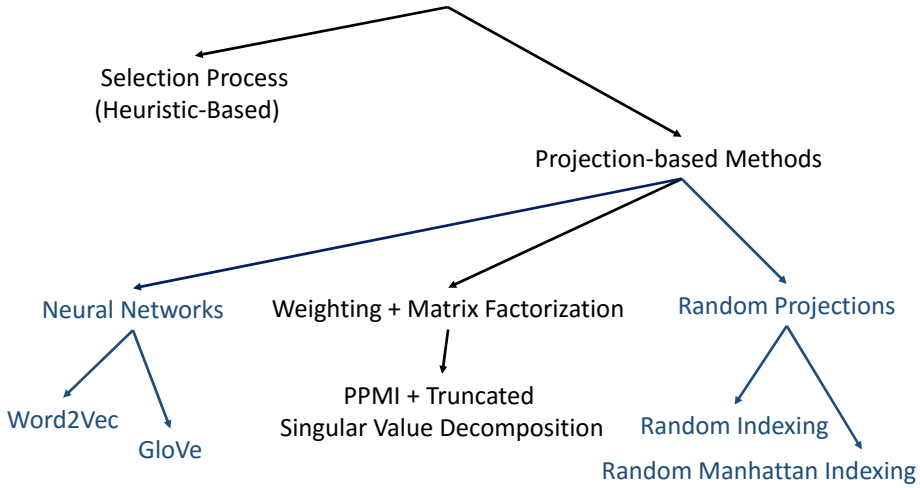
Full Stack of Processes



$$M'_{p\times m} = M_{p\times n} \times T_{n\times m}$$

20

$$M'_{p\times m} = M_{p\times n} \times T_{n\times m}$$



21

Limitations of Methods Based on Neural Nets and Matrix Factorization

- These methods are resource-intensive.
- Every time a model is update, $T_{n\times m}$ must be recomputed: neural-net must be retrained and SVD must be recomputed.
- During the training time models are not available.
- … .

× Not suitable for big-text data analytics.

× Not suitable for frequently updated text-data, e.g. blogs, tweeter data, etc.

22

Solution: Random Projections

- Methods based on *random projections* skip the computation of the transformation matrix using 'random' decisions.

- Often implemented as an incremental process:
  - Each context element is mapped to one **randomly generated *m***-dimensional *index vector* $\vec{r}$.
  - Each target entity (word) is mapped to an empty ***m***-dimensional context vector $\vec{v}$.
  - For each observed co-occurrence of an entity $\vec{v}$ and context element $\vec{r}$, do $\vec{v} = \vec{v} + \vec{r}$.

23

Random Indexing (RI)

- In RI, index vectors have asymptotic standard Gaussian distribution (i.e., $T_{n\times m} \sim \mathrm{N}(0,1)$).
- In RI, Elements $\mathrm{r_{ij}}$ of index vectors have the following distribution:

$$\mathrm{r_{ij}} = \begin{cases} +1 & \text{With probability } \frac{s}{2} \\ 0 & \text{With probability } 1-s \\ -1 & \text{With probability } \frac{s}{2} \end{cases},$$

  where s is a small number.
- Similarity Estimator: Cosine or other similarity measures in Euclidean spaces.

24

Random Manhattan Indexing (RMI)

- In RMI, index vectors have asymptotic standard *Cauchy* distribution (i.e., $T_{n\times m}\sim C(0,1)$).
- In RMI, Elements $\mathrm{r_{ij}}$ of index vectors have the following distribution:

$$\mathrm{r_{ij}} = \begin{cases} \left\lfloor \frac{1}{\mathrm{U}} \right\rfloor & \text{With probability } \frac{s}{2} \\ 0 & \text{With probability } 1-s \\ \left\lfloor \frac{-1}{\mathrm{U}} \right\rfloor & \text{With probability } \frac{s}{2} \end{cases},$$

where s is a small number and U is an i.i.d. random variable in (0,1).

- Estimator: The geometric mean to estimate Manhattan distances.

25

Word Embedding: Example

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

26

Word Embedding: Example

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

the taste of is best that I know

[0, 0 , 0 , 0 , 0 , 0 , 0 , 0]

Classic Model

Word Embedding: Example

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

the taste of is best that I know

$\overrightarrow{cherry}$ = [2 , 2 , 1 , 1 , 1 , 1 , 1 , 1]

28

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

Context Vector(cherry) = [0, 0 , 0 , 0]

Random indexing

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

Context Vector(cherry) = [0, 0 , 0 , 0]

Index Vector**(the) = [-1, 0, 0, 1]**
Index Vector**(taste) = [0, 1, 0,-1]**
Index Vector**(of) = [-1, 1, 0, 0]**
Index Vector**(is) = [0, 0, 1,-1]**
Index Vector**(best)= [1, 0,-1, 0]**
Index Vector**(that)= [0,-1, 1, 0]**
Index Vector**(I) = [0, 1,-1, 0]**
Index Vector**(know) = [-1, 0, 1, 0]**

30

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

Index Vector**(the) = [-1, 0, 0, 1]**
Index Vector**(taste) = [0, 1, 0,-1]**
Index Vector**(of) = [-1, 1, 0, 0]**
Index Vector**(is) = [0, 0, 1,-1]**
Index Vector**(the) = [-1, 0, 0, 1]**
Index Vector**(best)= [1, 0,-1, 0]**
Index Vector**(taste) = [0, 1, 0,-1]**
Index Vector**(that)= [0,-1, 1, 0]**
Index Vector**(I) = [0, 1,-1, 0]**
Index Vector**(know) = [-1, 0, 1, 0]**

$\sum\rightarrow$

31

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

Index Vector**(the) = [-1, 0, 0, 1]**
Index Vector**(taste) = [0, 1, 0,-1]**
Index Vector**(of) = [-1, 1, 0, 0]**
Index Vector**(is) = [0, 0, 1,-1]**
Index Vector**(the) = [-1, 0, 0, 1]**
Index Vector**(best)= [1, 0,-1, 0]**
Index Vector**(taste) = [0, 1, 0,-1]**
Index Vector**(that)= [0,-1, 1, 0]**
Index Vector**(I) = [0, 1,-1, 0]**
Index Vector**(know) = [-1, 0, 1, 0]**

$\sum\rightarrow$

Context Vector(cherry)=

[-3, 3, 1,-1]

32

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the best taste that I know.

Cherry = [-3, 3, 1,-1]

Cherry = [2, 2 , 1 , 1 , 1 , 1 , 1 , 1]

33

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word “cherry”.

The taste of **cherry** is the **very** best taste that I know.

	the	taste	of	is	best	that	I	know	very
Vector(cherry) =	[2,	2 ,	1 ,	1 ,	1 ,	1 ,	1 ,	1 ,	1]

34

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word "cherry".

The taste of **cherry** is the **very** best taste that I know.

Index Vector**(the)** = [-1, 0, 0, 1]
Index Vector**(taste)** = [0, 1, 0,-1]
Index Vector**(of)** = [-1, 1, 0, 0]
Index Vector**(is)** = [0, 0, 1,-1]
Index Vector**(the)** = [-1, 0, 0, 1]
Index Vector**(best)** = [1, 0,-1, 0]
Index Vector**(taste)** = [0, 1, 0,-1]
Index Vector**(that)** = [0,-1, 1, 0]
Index Vector**(I)** = [0, 1,-1, 0]
Index Vector**(know)** = [-1, 0, 1, 0]
Index Vector**(very)** = [0, 0, -1,1]

$\sum \rightarrow$ **Context Vector(cherry)=**

[-3, 3, 0, 0]

35

Random Projection Methods (RI Example)

Assume we want to built a vector representation for the word "cherry".

The taste of **cherry** is the **very** best taste that I know.

More index vectors

Index Vector**(the)** = [-1, 0, 0, 1]
Index Vector**(taste)** = [0, 1, 0,-1]
Index Vector**(of)** = [-1, 1, 0, 0]
Index Vector**(is)** = [0, 0, 1,-1]
Index Vector**(the)** = [-1, 0, 0, 1]
Index Vector**(best)** = [1, 0,-1, 0]
Index Vector**(taste)** = [0, 1, 0,-1]
Index Vector**(that)** = [0,-1, 1, 0]
Index Vector**(I)** = [0, 1,-1, 0]
Index Vector**(know)** = [-1, 0, 1, 0]
Index Vector**(very)** = [0, 0, -1,1]

$\sum \rightarrow$ **Context Vector(cherry)=**

[-3, 3, 0, 0]

FIXED DIMENSION

36

What is Wrong With RI then?

- RI, RMI, etc., are methods that **preserve distances between vectors**.
- If in the original n-dimensional space, $\boldsymbol{dist}(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) = \boldsymbol{\alpha}$, then in a randomly-projected models, with a high probability $\boldsymbol{p}$, $\boldsymbol{dist}'(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) = \boldsymbol{\alpha} \pm \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a small number:
  - For example, RI preserves Euclidean distances.
- Due to the absence of weighting, these distance measures do not show an acceptable performance in semantic similarity assessments.

37

What is Worse!?

- (RI, RMI, et.)-constructed models cannot be weighted after their construction:
  - The sum of row and column vectors is always zero!
  - For instance, given $\mathrm{PMI} = \log(\frac{v_{xy} \times \sum_{i=1}^{p} \sum_{j=1}^{n} v_{ij}}{\sum_{i=1}^{p} v_{iy} \times \sum_{j=1}^{n} v_{xj}})$, the PMI of the components of an RI model is always $^{0}/_{0}$.
- In best case, weighting must be done prior/during the construction of RI models.
  - Side Note: Due to Log-transformation, RI cannot be used for reducing the dimensionality of PPMI-weighted models (at least theoretically)!

38

What is desirable?

- A method that is as simple as RI.
- A method that can be used to construct word vectors incrementally.
- Resulting embedding must show an acceptable performance in semantic similarity assessments.
- Resulting embedding must be able to undergo weighting ***after*** their construction.

39

The Method: Positve-only Projections (PoP)

40

The Positive-only Projection Method (PoP)

- PoP is an incremental method that use index vectors with the following distribution:

$$r_{ij} = \begin{cases} \left\lfloor \frac{1}{\sqrt{\mathrm{U}}} \right\rfloor & \text{With probability } s \\ 0 & \text{With probability } 1 - s \end{cases}$$

  - where s is a small number such that at least one of the components of index vectors is a non-zero value.
  - U is an i.i.d. random variable in (0,1).
- In PoP-constructed models, similarities are computed using a correlation measure:
  - For unweighted models: Kendall's τ_b.
  - For PPMI-weighted models: Pearson's correlation r .

41

PoP Example

Assume we want to built a vector representation for the word "cherry".

The taste of **cherry** is the best taste that I know.

Index Vector**(the)** = **[0, 0, 0, 1]**
Index Vector**(taste)** = **[0, 2, 0, 0]**
Index Vector**(of)** = **[0, 1, 0, 0]**
Index Vector**(is)** = **[0, 0, 3, 0]**
Index Vector**(the)** = **[0, 0, 0, 1]**
Index Vector**(best)**= **[1, 0, 0, 0]**
Index Vector**(taste)** = **[0, 2, 0, 0]**
Index Vector**(that)**= **[0, 3, 0, 0]**
Index Vector**(I)** = **[0, 0, 9, 0]**
Index Vector**(know)** = **[0, 0, 8, 0]**

$\sum\rightarrow$ **Context Vector(cherry)=**

[1, 8, 20, 2]

42

Advantages of PoP

- In PoP, the sum of coordinates in index vectors and thus context vectors is always greater than zero.
  - PoP context vectors can be weighted using association measures such as PPMI **after** their construction:
    - Discriminative models can be built using small amount of computations and in an incremental fashion.

43

Experiments and Results

44

Evaluation: Methodology

- **The MEN relatedness test** consists of 3000 pairs of words.
  - Using a scale of 0 to 50, each pair is rated as "semantically related" by human.
  - Human ratings are used to obtain a ranked list of word pairs.
    - sun-n sunlight-n 50.000000
    - automobile-n car-n 50.000000
    - river-n water-n 49.000000
    - ...
    - cafe-n frog-n 4.000000
    - bakery-n zebra-n 0.000000

45

Evaluation: Methodology

- Given an embedding algorithm:
  - Compute similarities between word pairs.
  - Use the obtained similarities to rank the list of word pairs.
  - Compare the human ranked list with the list that is ranked by the algorithm (Spearman's rank correlation).
- Similarity between these two rankings is assumed as the figure of merit (i.e., the Spearman's rank correlation).

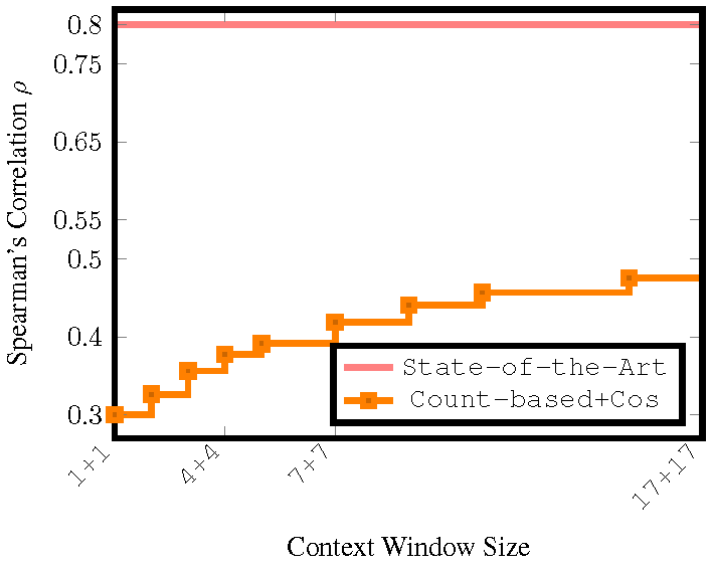
46

Evaluation: Methodology

- Note that for randomized algorithms such as RI and PoP, we repeat the ranking process 10 times and report the average of the observed performances (i.e., Spearman's correlations).
- We use the UKWaC corpus for collecting co-occurrence frequencies.
  - Only whitespace tokenization;
  - No part-of-speech tag;
  - No stop-word removal, etc.

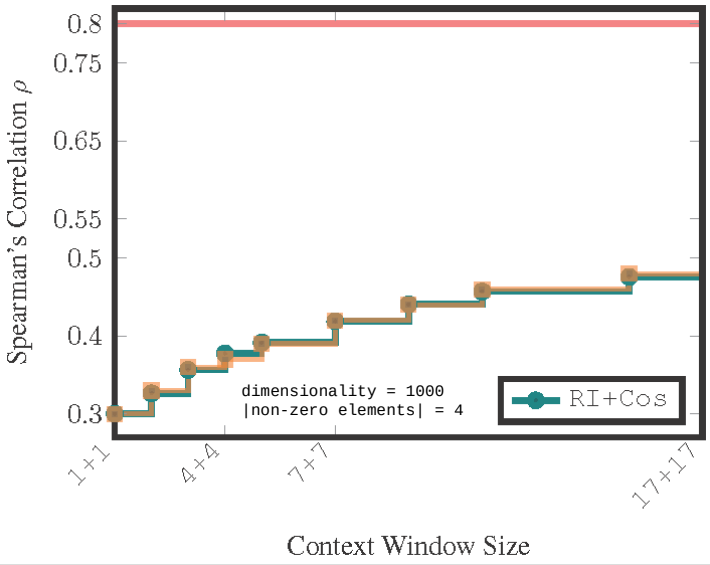
47

Results: Baselines



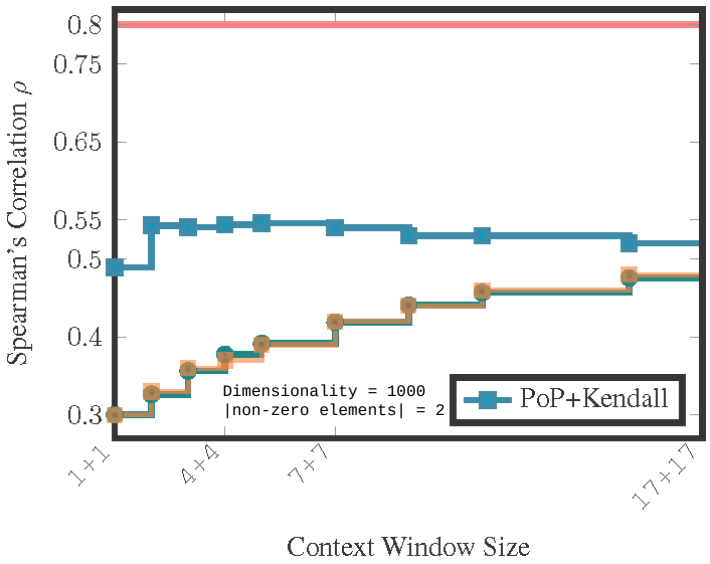
48

Results: Random Indexing



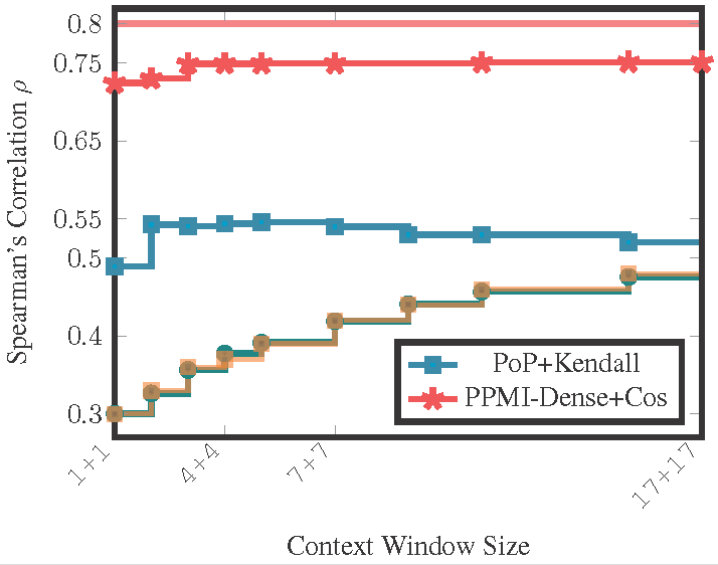
49

Results: PoP+Kendall



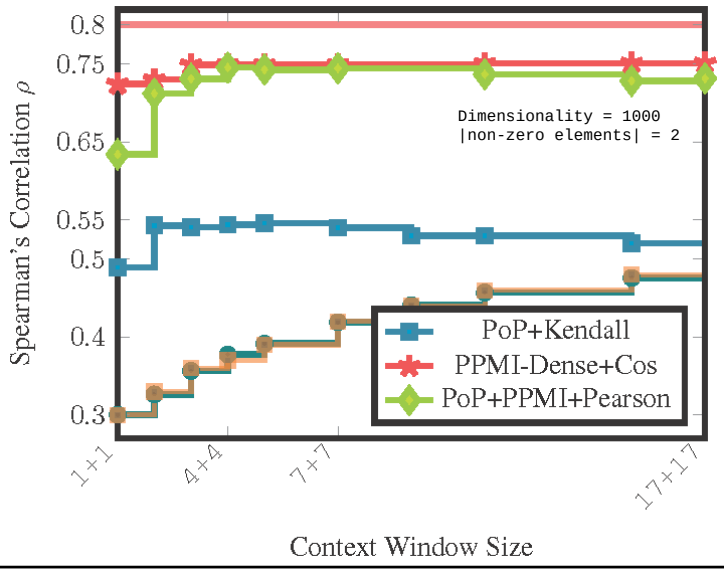
50

Results: PPMI Weighting



51

Results: PoP+PPMI+Pearson



52

Evaluation: PoP's Parameters

- PoP has two free parameters:
  - Dimensionality of vectors;
  - Number of non-zero elements.

53

Evaluation: PoP's Parameters

- In distance-preserving methods, such as RI, where the methods estimate distances by $\boldsymbol{dist}'(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) = \boldsymbol{\alpha} \pm \boldsymbol{\epsilon}$, these parameters determine the value of $\boldsymbol{\epsilon}$ and the probability $\boldsymbol{p}$ of its occurrence.

- PoP does not aim to preserve pairwise distances:
  - We still do not have a theoretical account for the effect of these parameters!

54

Evaluation: PoP's Parameters

- From a computational perspective, For both of these parameters, we prefer a small number.
- Lower dimensionality = Smaller memory/space complexity.
- Lower dimensionality = Faster weighting and similarity computation.
- Lower number of non-zero elements = Faster model construction.
- Note the when building PoP models, computational complexity is independent of its dimensionality.

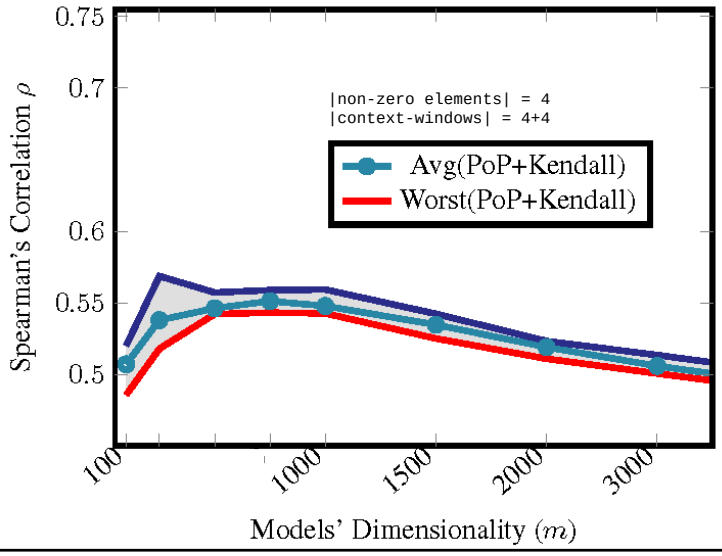
55

Evaluation: PoP's Parameters

- PoP model can be seen as a memory machine. We expect that:
  - The higher the dimensionality of a model, the higher its capacity to remember.
  - The higher the dimensionality of a model, the higher its sensitivity to (dis)similarities.
- The above means:
  - The higher the dimensionality of vectors, the better the performance of the method.
  - And smaller variance in results in each independent run of the method (i.e., smaller ϵ with higher p).

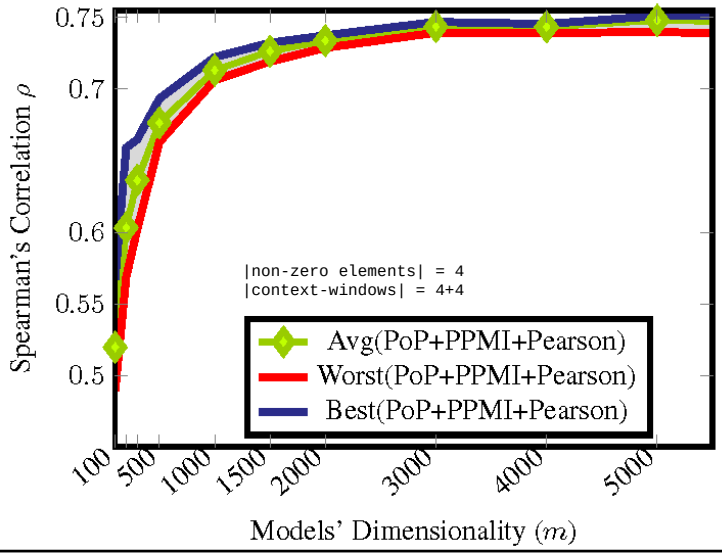
56

Results: Dimensionality



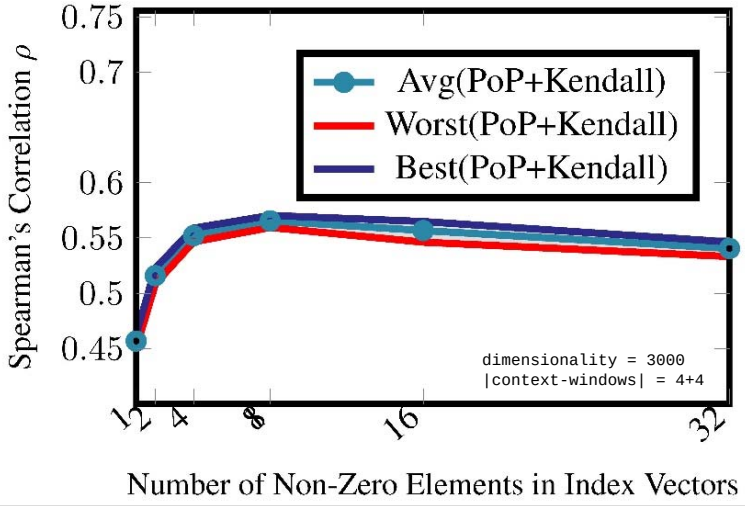
57

Results: Dimensionality



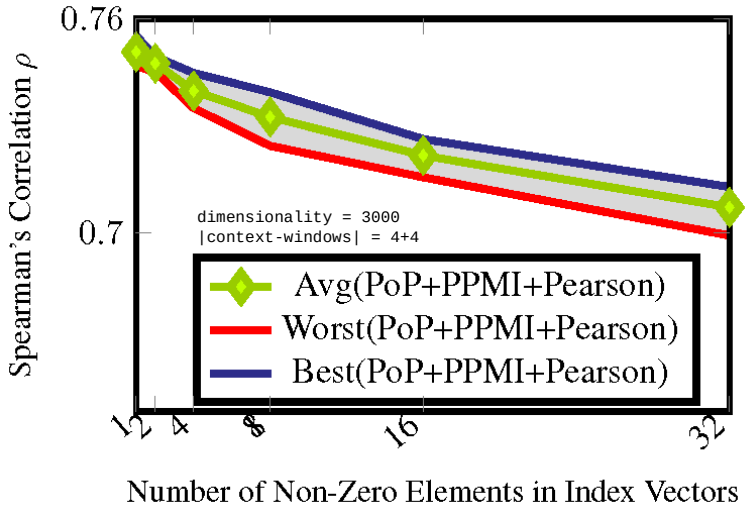
58

Results:
Number of Non-zero Elements



59

Results:
Number of Non-zero Elements



60

Evaluation: PoP's Parameters

- In short:
  - If possible, use vectors of higher dimensionality:
    - You can always truncate large vectors to small ones by simply ignoring some of the dimensions.
  - Use only one non-zero element, it is faster.
    - Every additional non-zero elements requires an additional summation per co-occurrence!

61

Results Beyond This Paper!

62

Task Name	Word pairs	Reference
MEN	3000	Bruni et. al, 2012
MTurk-287	287	Radinsky et. al, 2011
MTurk-771	771	Halawi and Dror, 2012
YP-130	130	Yang and Powers, 2006
SimLex-999	999	Hill et. al, 2014
Verb-144	144	Baker et. al, 2014
TOEFL	80	Landauer and Dumais, 1997
WS-353	353	Finkelstein et. al, 2002
WS-353-SIM	203	Agirre et. al, 2009
WS-353-REL	252	Agirre et. al, 2009
MC-30	30	Miller and Charles, 1930
RG-65	65	R and G, 1965
Rare-Word	2034	Luong et. al, 2013

63

Test → Model↓	YP-130	RW	MTURK-287	MTURK-771	MC-30	V-143	MEN-N	WS353-C	RG-65	TOEFL	WS353-S	WS353-R	SimLex999	Arithmetic Mean
Word2Vec (Baroni et. al., 2014)	0.50	0.32	0.66	**0.71**	**0.81**	**0.46**	**0.79**	0.63	**0.83**	0.91	0.75	0.52	0.46	**0.64**
Word2Vec (Mikolov et. al., 2013)*	0.40	0.38	**0.67**	0.57	0.62	0.27	0.64	0.64	0.54	NA	0.69	0.61	0.31	0.53
Senna (Collobert et. al., 2011)*	0.16	0.39	0.59	0.5	0.57	0.35	0.57	0.49	0.5	NA	0.61	0.4	0.27	0.45
PPP-777-w5+5 Wacky	**0.51**	**0.42**	0.6	0.62	0.76	0.29	0.73	0.69	0.74	0.88	0.74	0.64	0.37	0.61
PPP-777-w13+13 Wacky	0.49	0.40	**0.67**	0.63	0.77	0.27	0.75	**0.73**	0.76	0.83	**0.77**	**0.70**	0.32	0.62

64

*** Results for these vectors are taken from http://www.wordvectors.org (Faruqui and Dyer, 2014).**

Test → Model↓	YP-130	RW	MTURK-287	MTURK-771	MC-30	V-143	MEN-N	WS353-C	RG-65	TOEFL	WS353-S	WS353-R	SimLex999	Arithmetic Mean
Word2Vec (Baroni et. al., 2014)	0.50	0.32	0.66	**0.71**	**0.81**	**0.46**	**0.79**	0.63	**0.83**	0.91	0.75	0.52	0.46	0.64
Word2Vec (Mikolov et. al., 2013)*	0.40	0.38	0.67	0.57	0.62	0.27	0.64	0.64	0.54	NA	0.69	0.61	0.31	0.53
Senna (Collobert et. al., 2011)*	0.16	0.39	0.59	0.5	0.57	0.35	0.57	0.49	0.5	NA	0.61	0.4	0.27	0.45
PPP-777-w5+5 Wacky	0.51	**0.42**	0.6	0.62	0.76	0.29	0.73	0.69	0.74	0.88	0.74	0.64	0.37	0.61
PPP-777-w13+13 Wacky	0.49	0.40	0.67	0.63	0.77	0.27	0.75	**0.73**	0.76	0.83	**0.77**	**0.70**	0.32	0.62
PPP-777-w5+5 Wacky+FN+WN	**0.69**	0.36	0.65	0.65	**0.81**	0.21	0.71	0.67	0.81	**0.95**	0.77	0.56	**0.50**	0.64
PPP-777-Add-Models Only-Corpus!	**0.69**	0.39	**0.69**	0.67	0.80	0.22	0.74	**0.73**	0.77	**0.95**	**0.78**	0.64	0.49	**0.66**

65

*** Results for these vectors are taken from http://www.wordvectors.org (Faruqui and Dyer, 2014).**

Summary

- PoP is a new PPMI-enabled incremental method for constructing semantic spaces.
  - PoP is fast!
    - For instance, under a similar condition, PoP is almost twice as fast as Word2Vec.
  - PoP is easy to use!
    - Can be easily extended to models other than word spaces.
  - PoP allows "granular embedding"!
- Download basic codes (Python or Java) from https://user.phil-fak.uni-duesseldorf.de/~zadeh/material/pop-vectors/.
- Future Research:
  - De-randomization!
    - E.g., replacing random decisions with ones that are informed linguistically.

66