Embeddings Learned By Matrix Factorization

Hinrich Schütze

Center for Information and Language Processing, LMU Munich

2017-07-18

Overview

Outline

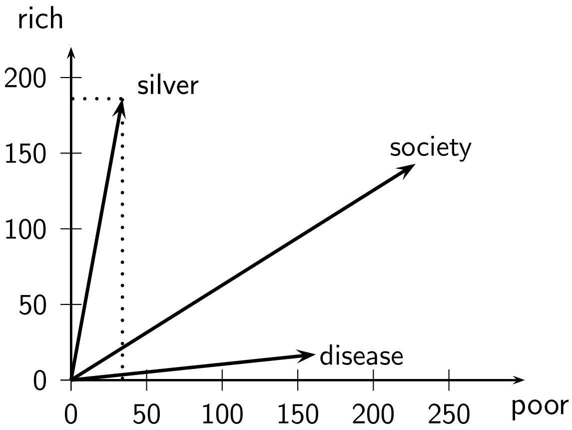
Embeddings

Definition

The embedding of a word w is a dense vector $\vec{v}(w) \in \mathcal{R}^k$ that represents semantic and other properties of w. Typical values are $50 \leq k \leq 1000$.

- In this respect, there is no difference to WordSpace: Both embeddings and WordSpace vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than WordSpace vectors.
- WordSpace vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

WordSpace


rich
200
150
100
50
0
silver
society
disease
0
50
100
150
200
250
poor

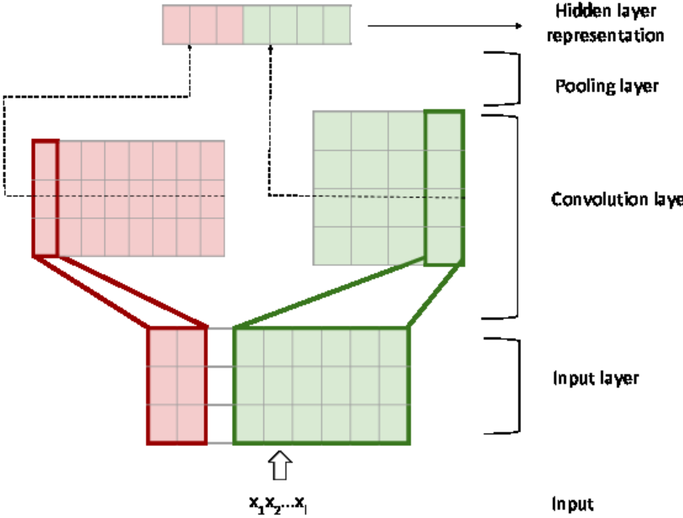
Word representations: Density and dimensionality

- WordSpace vectors are sparse and high-dimensional.
- In contrast, embeddings are dense and lower-dimensional.
- Why are embeddings potentially better?
- Embeddings are more efficient.
- Embeddings are often more effective.

Efficiency of embeddings

High dimensionality → slow training

The time to train a neural network is roughly linear in the dimensionality of word vectors.



WordSpace vectors: Example for non-effectiveness

- Example: polarity classification
- Cooccurrence with “bad” indicates negative polarity.
- But corpora are often random and noisy and a negative word may not have occurred with “bad”.
- Possible result:
  Incorrect classification based on WordSpace vectors
- Embeddings are more robust and “fill out” missing data.
- Details: below

Effectiveness of embeddings: Polarity

$$\vec{v}(\text{skunk}) = \begin{pmatrix} \text{high:0} \\ \text{cold:0} \\ \text{tree:1} \\ \text{good:0} \\ \text{white:1} \\ \text{bad:0} \\ \text{sweet:0} \\ \text{smelly:5} \end{pmatrix}$$

→

positive

neutral

negative

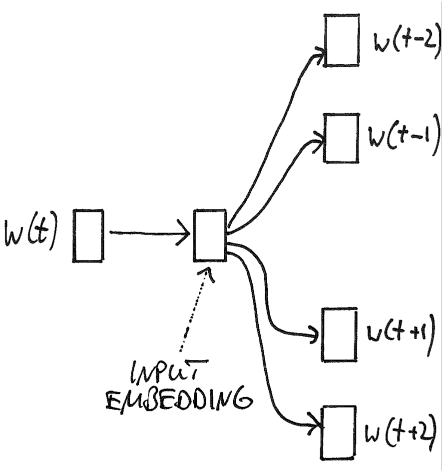
Effectiveness of WordSpace: Thought experiment

- Construct an example of a corpus and two words w_1 and w_2 occurring in it having the following properties:
  - w_1 and w_2 are semantically related.
  - The WordSpace vectors of w_1 and w_2 are not similar.
- Goal: Embeddings eliminate failure modes of WordSpace.

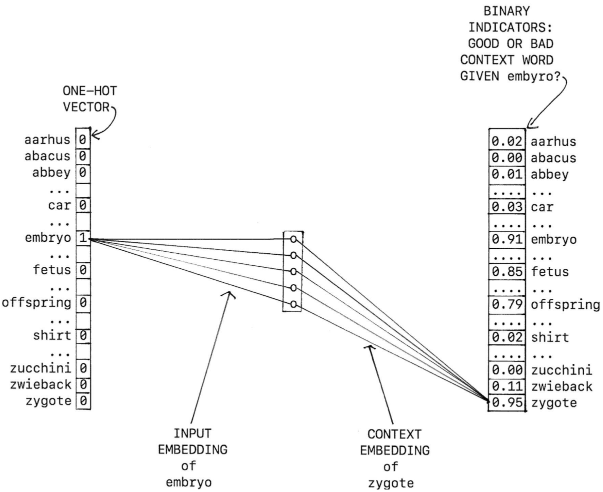
Best-known embedding model: word2vec skipgram

- word2vec skipgram is
  - more effective than WordSpace
    (embeddings and similarities of higher quality)
  - more efficient than WordSpace
    (lower dimensionality)

word2vec skipgram
predict, based on input word, a context word



word2vec skipgram
predict, based on input word, a context word



word2vec
learning = parameter estimation

- The embedding of a word is a real-valued vector $\in \mathcal{R}^k$.
- The coordinates are
  parameters that we need to learn/estimate from the corpus.
- Learning of a WordSpace model:
  (i) count, then (ii) PPMI weighting
- For word2vec, learning is more complicated.
- Two different methods
  - Embeddings learned via matrix factorization
  - Embeddings learned via gradient descent
- These estimation methods are roughly equivalent.

Example of an embedding vector: The numbers (or coordinates) are the parameters.

embedding of the work "skunk":

(-0.17823, -0.64124, 0.55163, -1.2453, -0.85144, 0.14677, 0.55626, -0.22915, -0.051651, 0.22749, 0.13377, -0.31821, 0.2266, -0.056929, -0.17589, -0.077204, -0.093363, 1.2414, -0.30274, -0.32308, 0.29967, -0.0098437, -0.411, 0.4479, 0.60529, -0.28617, 0.14015, 0.055757, -0.47573, 0.093785, -0.36058, -0.75834, -0.37557, -0.32435, -0.39122, -0.24014, 0.5508, -0.26339, 0.30862, 0.36182, 0.25648, 0.10642, -0.098591, -0.042246, 0.11275, 0.068252, 0.092793, -0.12239, 0.054094, 0.648, 0.30679, -0.38904, 0.32872, -0.22128, -0.26158, 0.48044, 0.86676, 0.1675, -0.37277, -0.53049, -0.13059, -0.076587, 0.22186, -0.81231, -0.2856, 0.20166, -0.41941, -0.60823, 0.66289, -0.059475, -0.14329, 0.0091092, -0.52114, -0.31488, -0.48999, 0.77458, -0.026237, 0.094321, -0.50531, 0.19534, -0.33732, -0.073171, -0.16321, 0.44695, -0.64077, -0.32699, -0.61268, -0.48275, -0.19378, -0.25791, 0.014448, 0.44468, -0.42305, -0.24903, -0.010524, -0.26184, -0.25618, 0.022102, -0.81199, 0.54065)

word2vec parameter estimation: Historical development vs. presentation in this lecture

- Mikolov et al. (2013) introduce word2vec, estimating parameters by gradient descent.
  - Still the learning algorithm used by default and in most cases
- Levy and Goldberg (2014) show near-equivalence to a particular type of matrix factorization.
  - Important because it links two important bodies of research: neural networks and distributional semantics
- More natural progression in this lecture:
  distributional semantics
  $\rightarrow$ embedding learning via matrix factorization
  $\rightarrow$ embedding learning via gradient descent

word2vec skipgram: Embeddings learned via gradient descent

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

word2vec skipgram: Embeddings learned via matrix factorization

Neural Word Embedding as Implicit Matrix Factorization

Omer Levy
Department of Computer Science
Bar-Ilan University
omerlevy@gmail.com

Yoav Goldberg
Department of Computer Science
Bar-Ilan University
yoav.goldberg@gmail.com

Abstract

We analyze skip-gram with negative-sampling (SGNS), a word embedding method introduced by Mikolov et al., and show that it is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs (shifted by a global constant). We find that another embedding method, NCE, is implicitly factorizing a similar matrix, where each cell is the (shifted) log conditional probability of a word given its context.

Outline

Dot product / scalar product

$$\vec{w}\vec{c} = \sum_i w_i c_i$$

Example:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = w_1 c_1 + w_2 c_2 + w_3 c_3$$

Linear algebra review: $C = AB$

			B	
			1	-2
			-1	2
A	1	1	0	0
	2	1	1	-2

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

Linear algebra review: $C = AB$

				B
			0.7	0.3
			0.2	0.8
	0	1	0.2	0.8
A	0.2	0.8	0.3	0.7
	0.3	0.7	0.35	0.65

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$
$$C_{31} = A_{31}B_{11} + A_{32}B_{21}$$
$$C_{32} = A_{31}B_{12} + A_{32}B_{22}$$

Euclidean length of a vector $\vec{d}$

$$|\vec{d}| = \sqrt{\sum_{i=1}^{n} d_i^2}$$

$\vec{c}$ and $\vec{d}$ are orthogonal iff

$$\sum_{i=1}^{n} c_i \cdot d_i = 0$$

Exercise

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

Show: column d_1 has unit length: $\sqrt{\sum_i d_{i1}^2} = 1$

Show: columns d_1, d_2 are orthogonal: $\sum_i d_{i1} \cdot d_{i2} = 0$

$0.75^2 + 0.29^2 + 0.28^2 + 0.00^2 + 0.53^2 = 1.0059$

$-0.75 * -0.28 + -0.29 * -0.53 + 0.28 * -0.75 + 0.00 * 0.00 + -0.53 * 0.29 = 0$

Outline

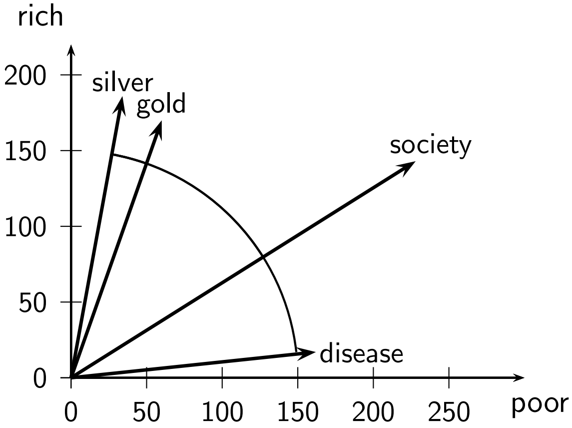
Outline of this section

- Recall:
  We learn embedding parameters by matrix factorization.
- We need an input matrix for matrix factorization.
- Brief recap on how to create the input matrix
- (You already know this,
  but it looks slightly different.)
- Also: link to information retrieval
- This type of “technology” comes from information retrieval.
- Brief overview of information retrieval setting

Vector representations: Words vs. Documents/Queries

- Statistical NLP & Deep learning:
  Embeddings as model for word similarity
- Information retrieval:
  Vector representations as model of query-document similarity
- Simple search engine:
  - User enters query.
  - Query is transformed into query vector.
  - Documents are transformed into document vectors.
  - Order document vectors according to similarity to query
  - Return ranked list of documents to user:
    The documents with highest similarity to query.

Basis for WordSpace: Cooccurrence→ Similarity



The similarity between two words is the cosine of the angle between them.

Small angle: silver and gold are similar. Medium-size angle: silver and society are not very similar. Large angle: silver and disease are

Documents ranked according to similarity to query

automobile prices

All News Shopping Images Maps More Settings Tools

About 69,500,000 results (0.41 seconds)

Automobile aus Deutschland - 2,4 Mio. Gebraucht- & Neuwagen
Ad www.autoscout24.de/auto/mobile
4.3 ★★★★★ rating for autoscout24.de
Jetzt schnell, einfach & unkompliziert Autos aller Marken in Ihrer Nähe finden.
Europaweite Angebote · Alle Fahrzeugdetails · Kostenlos verkaufen · Ausgezeichneter Service
Modelle: VW Turan, Kia Sportage, BMW X1, Audi A3

AutoScout24 Neuwagen	Neuwagen	Fabrikneue Autos
from **€8,000.00**	from **€10K**	from **€12.5K**
verschiedene Modelle	verschiedene Modelle	verschiedene Modelle

Kelley Blue Book - New and Used Car Price Values, Expert Car Reviews
https://www.kbb.com/
Check KBB **car price** values when buying and selling new or used vehicles. Recognized by consumers and the automotive industry since 1926.
Resale Value · Used Car Prices · New Cars · Motorcycles

NADAguides: New Car Prices and Used Car Book Values
https://www.nadaguides.com/
Research the latest new **car prices**, deals, used car values, specs and more. NADA Guides is the leader in accurate **vehicle pricing** and vehicle information.
New Car Prices & Used Car ... · Motorcycles · RV Prices and Values · Trucks

Words ranked according to similarity to query word

1.000 silver 0.865 bronze 0.842 gold 0.836 medal 0.826 medals 0.761 relay 0.740 medalist 0.737 coins 0.724 freestyle 0.720 metre 0.716 coin 0.714 copper 0.712 golden 0.706 event 0.701 won 0.700 foil 0.698 Winter 0.684 Pan 0.680 vault 0.675 jump

Setup for cooccurrence count matrix
Dimension words (w_2) and points/vectors (w_1)

		w_2				
		rich	poor	silver	society	disease
	rich					
	poor					
w_1	silver					
	society					
	disease					

Cooccurrence count (CC) matrix

		w_2 rich	poor	silver	society	disease
	rich	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$
	poor	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$
w_1	silver	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$
	society	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$
	disease	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$	$\mathrm{CC}(w_1, w_2)$

PPMI matrix C
This is the input to matrix factorization, which will compute word embeddings.

		w_2 rich	poor	silver	society	disease
	rich	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$
	poor	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$
w_1	silver	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$
	society	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$
	disease	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$	$\text{PPMI}(w_1, w_2)$

PPMI: Weighting of raw cooccurrence counts

- PMI: pointwise mutual information
- $$\text{PMI}(w, c) = \log \frac{P(wc)}{P(w)P(c)}$$
- PPMI = positive pointwise mutual information
- $\text{PPMI}(w, c) = \max(0, \text{PMI}(w, c))$
- More generally (with offset k): $\text{PPMI}(w, c) = \max(0, \text{PMI}(w, c) - \text{k})$

Information Retrieval: Word-document matrix

	doc 1	doc 2	doc 3	doc 4	doc 5	query
anthony	5.25	3.18	0.0	0.0	0.0	0.35
brutus	1.21	6.10	0.0	1.0	0.0	0.0
caesar	8.59	2.54	0.0	1.51	0.25	0.0
calpurnia	0.0	1.54	0.0	0.0	0.0	0.0
cleopatra	2.85	0.0	0.0	0.0	0.0	0.0
mercy	1.51	0.0	1.90	0.12	5.25	0.88
worser	1.37	0.0	0.11	4.15	0.25	0
...						

Outline

Matrix factorization: Overview

- We will decompose the word-document matrix into a product of matrices.
- The particular decomposition we'll use: singular value decomposition (SVD).
- SVD: $C = U\Sigma V^T$ (where C = word-document matrix)
- We will then use the SVD to compute a new, improved word-document matrix C'.
- We'll get better query-document similarity values out of C' (compared to C).
- Using SVD for this purpose is called latent semantic indexing or LSI. □

Matrix factorization: Embeddings

- We will decompose the cooccurrence matrix into a product of matrices.
- The particular decomposition we'll use: singular value decomposition (SVD).
- SVD: $C = U\Sigma V^T$ (where C = cooccurrence matrix)
- We will then use the SVD to compute a new, improved cooccurrence matrix C'.
- We'll get better word-word similarity values out of C' (compared to C).

Example of $C = U\Sigma V^T$: The matrix C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

We use a non-weighted matrix here to simplify the example. □

Example of $C = U\Sigma V^T$: The matrix U

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

One row per word, one column per $\min(M, N)$ where M is the number of words and N is the number of documents.

This is an orthonormal matrix: (i) Row vectors have unit length. (ii) Any two distinct row vectors are orthogonal to each other.

Think of the dimensions as "semantic" dimensions that capture distinct topics like politics, sports, economics. $2 = \text{land}/\text{water}$

Each number u_{ij} in the matrix indicates how strongly related word i is to the topic represented by semantic dimension j. □

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

Example of $C = U\Sigma V^T$: The matrix Σ

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

This is a square, diagonal matrix of dimensionality $\min(M, N) \times \min(M, N)$.

The diagonal consists of the singular values of C.

The magnitude of the singular value measures the importance of the corresponding semantic dimension.

We'll make use of this by omitting unimportant dimensions. □

Example of $C = U\Sigma V^T$: The matrix V^T

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

One column per document, one row per $\min(M, N)$ where M is the number of words and N is the number of documents.

Again: This is an orthonormal matrix: (i) Column vectors have unit length. (ii) Any two distinct column vectors are orthogonal to each other.

These are again the semantic dimensions from matrices U and Σ that capture distinct topics like politics, sports, economics.

Each number v_{ij} in the matrix indicates how strongly related document i is to the topic represented by semantic dimension

Example of $C = U\Sigma V^T$: All four matrices

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00

$\times$

Recall unreduced decomposition $C = U\Sigma V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00

$\times$

Exercise: Why can this be viewed as soft clustering?

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

$=$

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

$\times$

Σ	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	1.28	0.00	0.00

$\times$

SVD: Summary

- We've decomposed the word-document matrix C into a product of three matrices: $U\Sigma V^T$.
- The word matrix U – consists of one (row) vector for each word
- The document matrix V^T – consists of one (column) vector for each document
- The singular value matrix Σ – diagonal matrix with singular values, reflecting importance of each dimension
- Next: Why are we doing this?

Property of SVD that we exploit here

- Key property:
  Each singular value tells us how important its dimension is.
- By setting less important dimensions to zero, we keep the important information, but get rid of the "details".
- These details may
  - be noise – in that case, reduced SVD vectors are a better representation because they are less noisy.
  - make things dissimilar that should be similar – again, reduced SVD vectors are a better representation because they represent similarity better.
- Analogy for "fewer details is better"
  - Image of a blue flower
  - Image of a yellow flower
  - Omitting color makes is easier to see the similarity □

Reducing the dimensionality to 2

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

Actually, we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$. □

Reducing the dimensionality to 2

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

=

U	1	2	3	4	5
ship	−0.44	−0.30	0.57	0.58	0.25
boat	−0.13	−0.33	−0.59	0.00	0.73
ocean	−0.48	−0.51	−0.37	0.00	−0.61
wood	−0.70	0.35	0.15	−0.58	0.16
tree	−0.26	0.65	−0.41	0.58	−0.09

×

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

×

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.28	−0.75	0.45	−0.20	0.12	−0.33
4	0.00	0.00	0.58	0.00	−0.58	0.58
5	−0.53	0.29	0.63	0.19	0.41	−0.22

Original matrix C vs. reduced $C_2 = U\Sigma_2 V^T$

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

We can view C_2 as a two-dimensional representation of the matrix C. We have performed a dimensionality reduction to two dimensions.

Exercise

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Compute the similarity between d_2 and d_3 for the original matrix and for the reduced matrix.

Why the reduced matrix C_2 is better than C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

"boat" and "ship" are semantically similar. The "reduced" similarity measure reflects this.

What property of the SVD reduction is responsible for improved similarity? □

Why the reduced matrix C_2 is better than C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Similarity of d_2 and d_3 in the original space: 0.

Similarity of d_2 and d_3 in the reduced space: $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

Why the reduced matrix C_2 is better than C

C	d_1	d_2	d_3	d_4	d_5	d_6
ship	1	0	1	0	0	0
boat	0	1	0	0	0	0
ocean	1	1	0	0	0	0
wood	1	0	0	1	1	0
tree	0	0	0	1	0	1

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.85	0.52	0.28	0.13	0.21	−0.08
boat	0.36	0.36	0.16	−0.20	−0.02	−0.18
ocean	1.01	0.72	0.36	−0.04	0.16	−0.21
wood	0.97	0.12	0.20	1.03	0.62	0.41
tree	0.12	−0.39	−0.08	0.90	0.41	0.49

Similarity of d_2 and d_3 in the original space: 0.

Similarity of d_2 and d_3 in the reduced space: $0.52 * 0.28 + 0.36 * 0.16 + 0.72 * 0.36 + 0.12 * 0.20 + -0.39 * -0.08 \approx 0.52$

Exercise: Compute matrix product

C_2	d_1	d_2	d_3	d_4	d_5	d_6
ship	0.09	0.16	0.06	-0.19	-0.07	-0.12
boat	0.10	0.17	0.06	-0.21	-0.07	-0.14
ocean	0.15	0.27	0.10	-0.32	-0.11	-0.21
wood	-0.10	-0.19	-0.07	0.22	0.08	0.14
tree	-0.19	-0.34	-0.12	0.41	0.14	0.27

???????=

U	1	2	3	4	5
ship	-0.44	-0.30	0.57	0.58	0.25
boat	-0.13	-0.33	-0.59	0.00	0.73
ocean	-0.48	-0.51	-0.37	0.00	-0.61
wood	-0.70	0.35	0.15	-0.58	0.16
tree	-0.26	0.65	-0.41	0.58	-0.09

$\times$

Σ_2	1	2	3	4	5
1	0.00	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

$\times$

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	-0.75	-0.28	-0.20	-0.45	-0.33	-0.12
2	-0.29	-0.53	-0.19	0.63	0.22	0.41
3	0.28	-0.75	0.45	-0.20	0.12	-0.33
4	0.00	0.00	0.58	0.00	-0.58	0.58
5	-0.53	0.29	0.63	0.19	0.41	-0.22

word2vec learning via matrix factorization

- Collect and weight cooccurrence matrix
- Compute SVD of cooccurrence matrix
- Reduce the space
- embeddings = left singular vectors (left matrix)

embeddings = left singular vectors

U	1	2	3	4	5
ship	−0.44	−0.30	0.00	0.00	0.00
boat	−0.13	−0.33	0.00	0.00	0.00
ocean	−0.48	−0.51	0.00	0.00	0.00
wood	−0.70	0.35	0.00	0.00	0.00
tree	−0.26	0.65	0.00	0.00	0.00

Σ_2	1	2	3	4	5
1	2.16	0.00	0.00	0.00	0.00
2	0.00	1.59	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

V^T	d_1	d_2	d_3	d_4	d_5	d_6
1	−0.75	−0.28	−0.20	−0.45	−0.33	−0.12
2	−0.29	−0.53	−0.19	0.63	0.22	0.41
3	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00

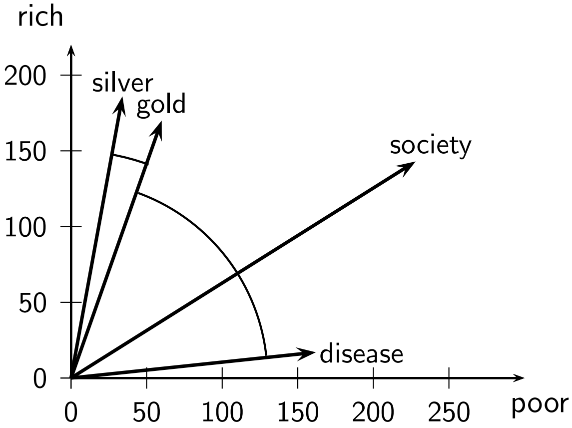
Actually, we only zero out singular values in Σ. This has the effect of setting the corresponding dimensions in U and V^T to zero when computing the product $C = U\Sigma V^T$. □

Outline

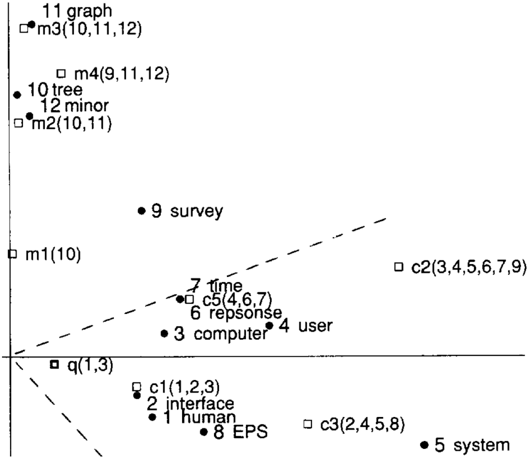
Optimality

- SVD is optimal in the following sense.
- Keeping the k largest singular values and setting all others to zero gives you the optimal approximation of the original matrix C. Eckart-Young theorem
- Optimal: no other matrix of the same rank (= with the same underlying dimensionality) approximates C better.
- Measure of approximation is Frobenius norm: $||C||_F = \sqrt{\sum_i \sum_j c_{ij}^2}$
- So SVD uses the "best possible" matrix.
- There is only one best possible matrix – unique solution (modulo signs).
- Caveat: There is only a tenuous relationship between the Frobenius norm and cosine similarity between documents. □

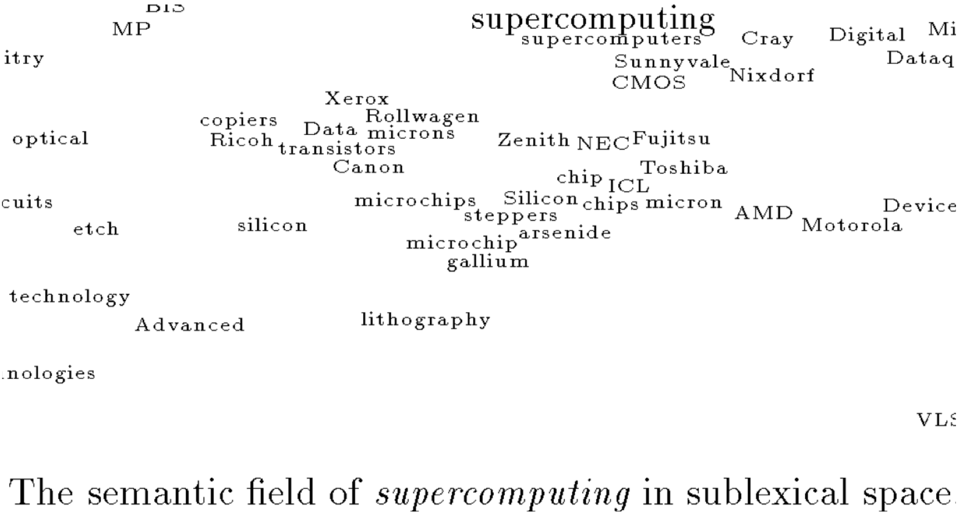
Embeddings (1): Vector space model (Salton, 1960s)



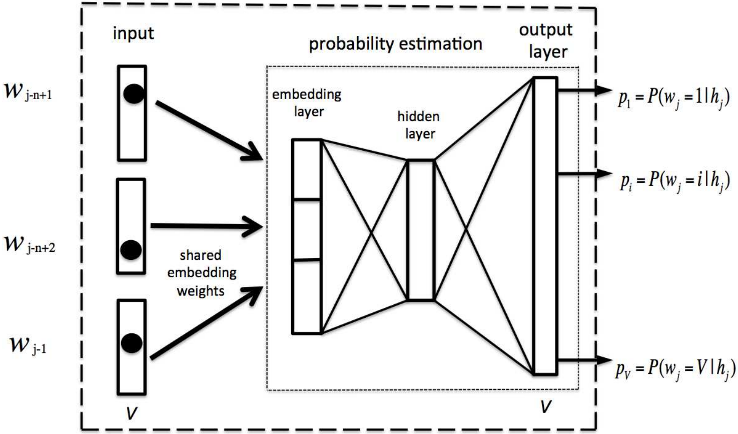
Embeddings (2): Latent Semantic Indexing (Deerwester, Dumais, Landauer . . . , 1980s)



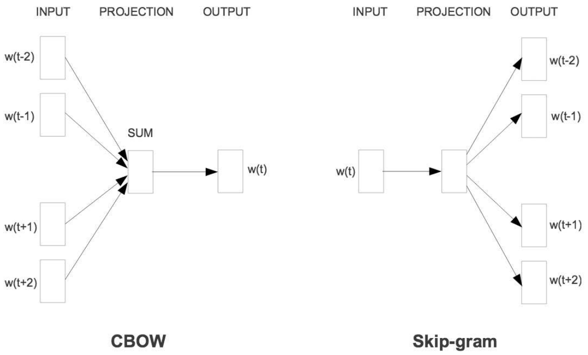
Embeddings (3): SVD-based methods (Schütze, 1992)



Embeddings (4): Neural models (Bengio, Schwenk, . . . , 2000s)



Embeddings (5): word2vec



Embeddings (6): SVD-based methods (Stratos et al., 2015)

SPECTRAL-TEMPLATE

Input: word-context co-occurrence counts $\#(w,c)$, dimension m, transformation method t, scaling method s, context smoothing exponent $\alpha \leq 1$, singular value exponent $\beta \leq 1$

Output: vector $v(w) \in \mathbb{R}^m$ for each word $w \in [n]$

Definitions: $\#(w) := \sum_c \#(w,c)$, $\#(c) := \sum_w \#(w,c)$, $N(\alpha) := \sum_c \#(c)^\alpha$

1. Transform all $\#(w,c)$, $\#(w)$, and $\#(c)$:

$$\#(\cdot) \leftarrow \begin{cases} \#(\cdot) & \text{if } t = \text{—} \\ \log(1+\#(\cdot)) & \text{if } t = \text{log} \\ \#(\cdot)^{2/3} & \text{if } t = \text{two-thirds} \\ \sqrt{\#(\cdot)} & \text{if } t = \text{sqrt} \end{cases}$$

2. Scale statistics to construct a matrix $\Omega \in \mathbb{R}^{n \times n}$:

$$\Omega_{w,c} \leftarrow \begin{cases} \#(w,c) & \text{if } s = \text{—} \\ \frac{\#(w,c)}{\#(w)} & \text{if } s = \text{reg} \\ \max\left(\log \frac{\#(w,c)N(\alpha)}{\#(w)\#(c)^\alpha}, 0\right) & \text{if } s = \text{ppmi} \\ \frac{\#(w,c)}{\sqrt{\#(w)\#(c)^\alpha}}\sqrt{\frac{N(\alpha)}{N(1)}} & \text{if } s = \text{cca} \end{cases}$$

Perform rank-m SVD on $\Omega \approx U\Sigma V^\top$ where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_m)$ is a diagonal matrix of ordered singular values $\sigma_1 \geq \cdots \geq \sigma_m \geq 0$.

Define $v(w) \in \mathbb{R}^m$ to be the w-th row of $U\Sigma^\beta$ normalized to have unit 2-norm.

Embeddings (7): GloVe (Pennington, Socher, Manning, 2014)

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2$$

Takeaway
Limitations of WordSpace

- WordSpace vectors can be inefficient.
  (large number of parameters when used in deep learning)
- WordSpace vectors can be ineffective.
  (due to randomness and noisiness of cooccurrence)

Takeaway
Definition of embedding

- Real-valued vector representation of word w
- Represents semantic and other properties of w
- Low dimensionality k (e.g., $50 \leq k \leq 1000$)
- Dense (as opposed to sparse)

Takeaway
Embeddings my matrix factorization

- Compute PPMI cooccurrence matrix
- Decompose it using SVD
- Reduce left matrix U to d dimensions
- Reduced U is then the embeddings matrix.

Resources

- Chapter 18 of IIR at `http://cislmu.org`
- Deerwester et al.'s paper on latent semantic indexing
- Paper on probabilistic LSI by Thomas Hofmann
- Neural Word Embeddings as Implicit Matrix Factorization. Omer Levy and Yoav Goldberg. NIPS 2014.

Outline

```
code/word2vec -train tinycorpus.txt
-size 50 -read-vocab voc.txt -output vec.txt

code/word2vec -train tinycorpus.txt
-size 50 -read-vocab voc.txt -binary 1 -output
vec.bin
```

GoogleNews embeddings

- Embedding basis for `cis.lmu.de/schuetze/e`
- These were computed with word2vec skipgram.
- 3,000,000 words and phrases
- 300-dimensional embeddings
- Trained on 100 billion word Google news dataset

word2vec parameters

- -train <file>
  input corpus
- -output <file>
  word vectors are saved in output file
- -size <int>
  dimensionality of word vectors
- -window <int>
  skip length between words
- -sample <float>
  threshold for downsampling frequent words
- -hs <int>
  use hierarchical softmax (0/1)
- -negative <int>
  number of negative samples
- -threads <int>
  number of threads

word2vec parameters

- -iter <int>
  number of training iterations
- -min-count <int>
  discard words that occur less often than this
- -alpha <float>
  starting learning rate
- -classes <int>
  output word classes rather than word vectors
- -debug <int>
  set the debug mode
- -binary <int>
  use binary or plain text output format (0/1)

word2vec parameters

- -save-vocab <file>
  save the vocabulary
- -read-vocab <file>
  read the vocabulary from file (don't take it from corpus)
- -cbow <int>
  use cbow or not (0/1)