Distributional Semantics
Word Embeddings

Hinrich Schütze

Center for Information and Language Processing, LMU Munich

2019-08-29

Overview

Outline

Key concept: Semantic similarity

- Two words are semantically similar
  if they have similar meanings.
- Examples of similar words:
  - "furze" $\leftrightarrow$ "gorse"
  - "astronaut" $\leftrightarrow$ "cosmonaut"
  - "banana" $\leftrightarrow$ "apple" (these two are less similar)
- Examples of not similar words:
  - "car" $\leftrightarrow$ "flower"
  - "car" $\leftrightarrow$ "pope"
- Examples of similar words that are not nouns:
  - "huge" $\leftrightarrow$ "large"
  - "eat" $\leftrightarrow$ "devour"

Furze = gorse = whin

Key concept: Semantic similarity

- Two words are semantically similar
  if they have similar meanings.
- Examples of similar words:
  - "furze" $\leftrightarrow$ "gorse"
  - "astronaut" $\leftrightarrow$ "cosmonaut"
  - "banana" $\leftrightarrow$ "apple" (these two are less similar)
- Examples of not similar words:
  - "car" $\leftrightarrow$ "flower"
  - "car" $\leftrightarrow$ "pope"
- Examples of similar words that are not nouns:
  - "huge" $\leftrightarrow$ "large"
  - "eat" $\leftrightarrow$ "devour"

Semantic relatedness

- Two words are semantically related
  if their meanings are related.
- Example: "car" $\leftrightarrow$ "motorway"
- A car is not similar to a motorway, but there is an obvious relationship between them.
- Linguistically / ontologically well defined relations: synonymy, antonymy, hypernymy, meronymy, troponymy, ...
- Note: "car" $\leftrightarrow$ "motorway" isn't an instance of any of these!
- More generally: Two words are semantically related if their meanings are related in the real world. For example, if one word describes a given situation ("I'm on the motorway"), then it is very likely that the other word also describes this situation ("I'm in a car").
- There is a spectrum here:
  synonymous, very similar, less similar, related, unrelated

Here: Similarity includes relatedness

- In what follows,
  I will use semantic similarity as a general term
  that includes semantic similarity and semantic relatedness.

Distributional semantics and word embeddings

- Distributional semantics is an approach to semantics that is based on the contexts of words in large corpora.
- The basic notion formalized in distributional semantics is semantic similarity.
- Word embeddings are the modern incarnation of distributional semantics – adapted to work well with deep learning.

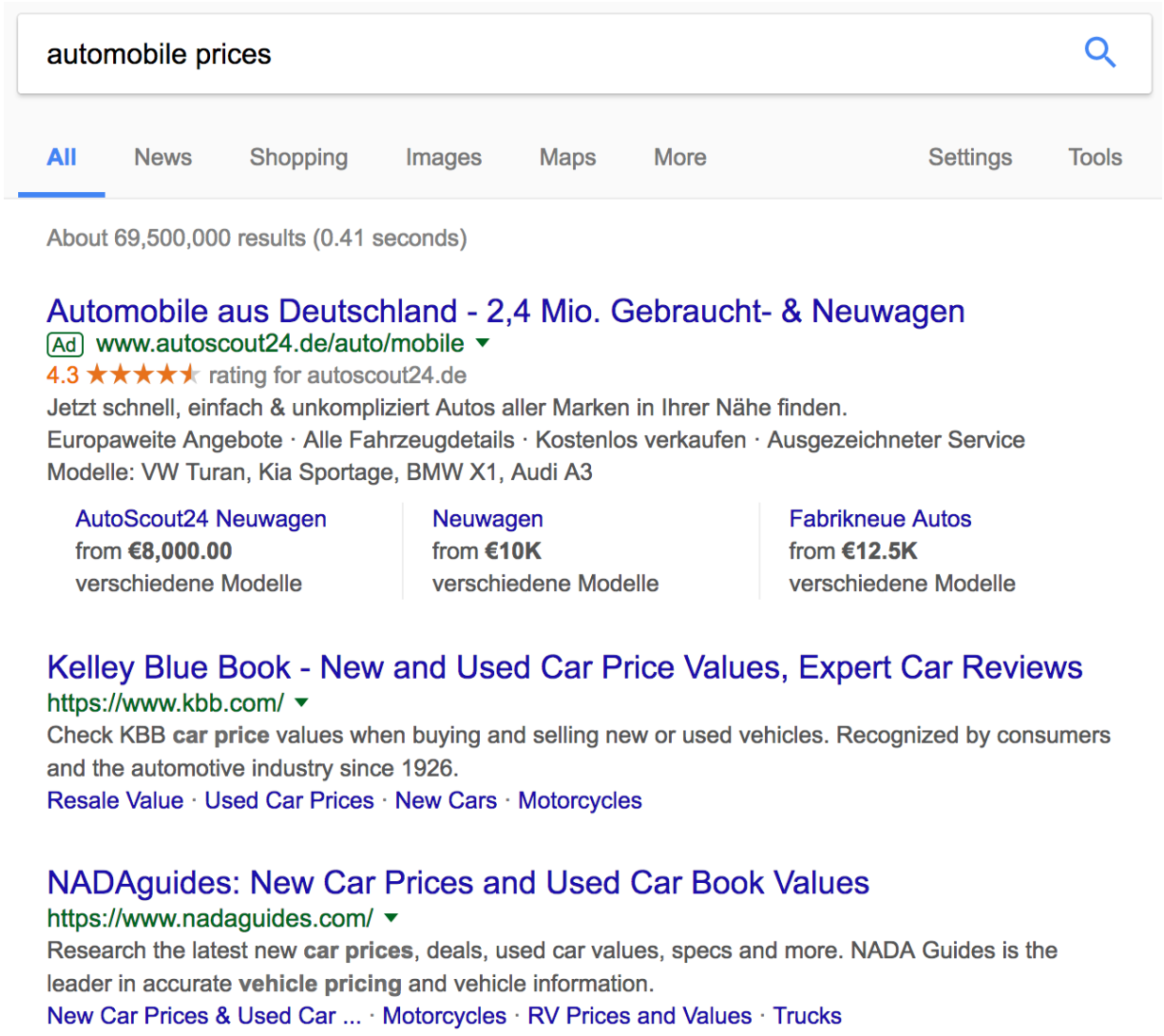
Why is semantic similarity interesting?

- It's a solvable problem (see below).
  - Many other things we want to do with language are more interesting, but much harder to solve.
- We do not need annotated data.
- There are many applications for semantic similarity.
- Two examples of applications
  - 1. Direct use of measures of semantic similarity
  - 2. OOVs, representations for unknown words

Application 1: Direct use of semantic similarity

- Query expansion in information retrieval
- User types in query [automobile]
- Search engine expands with semantically similar word [car]
- The search engine then uses the query [car OR automobile]
- Better results for the user

Google: Internal model of semantic similarity



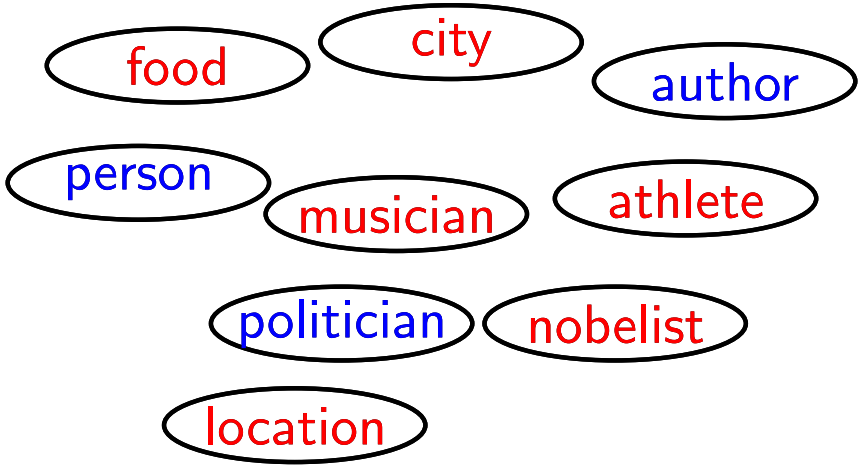
Application 2: OOVs, representations for unknown words

- Entity typing
- We often need to infer properties of a new (OOV) entity.
- For example, if the system encounters "Fonsorbes" for the first time, it is useful to be able to infer that it is a town.
- Embeddings contain valuable information about OOVs.

Embedding-based entity typing:
Given embedding, predict correct types of entity

$$\begin{pmatrix} +0.12 \\ +0.57 \\ -0.69 \\ +0.05 \\ +0.82 \\ +0.56 \\ -0.91 \\ +0.10 \end{pmatrix}$$

$\vec{v}(\text{Clinton})$



Cf. Wang, Zhang, Feng & Chen (2014), Yogatama, Gillick & Lazic (2015), Neelakantan & Chang (2015), Yaghoobzadeh & Schütze (2015, 2017)

Why is semantic similarity interesting?

- It's a solvable problem (see below).
  - Many other things we want to do with language are more interesting, but much harder to solve.
- We do not need annotated data.
- There are many applications for semantic similarity.
- Two examples of applications
  - 1. Direct use of measures of semantic similarity
  - 2. OOVs, representations for unknown words

Outline

Semantic similarity based on cooccurrence

- Assume the equivalence of:
  - Two words are semantically similar.
  - Two words occur in similar contexts
  - Two words have similar word neighbors in the corpus.
- Elements of this are from: Leibniz, Firth, Harris, Miller
- Strictly speaking, similarity of neighbors is neither necessary nor sufficient for semantic similarity.
- But we will see that results are good.

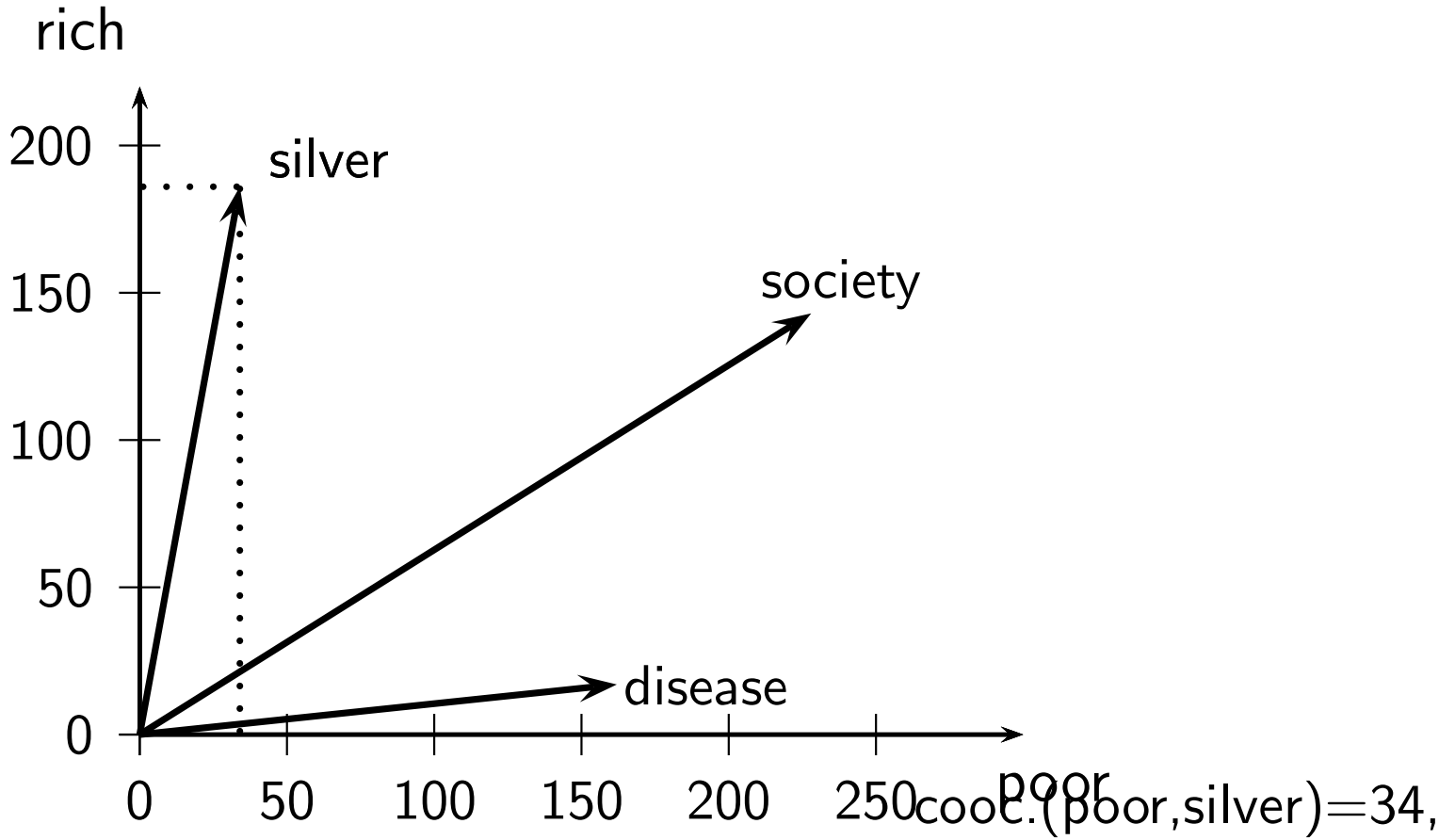
Key concept: Cooccurrence count

- Cooccurrence count:
  basis for precise definition of "semantic similarity"
- The cooccurrence count of words w_1 and w_2 in a corpus is the number of times that w_1 and w_2 cooccur.
- Different definitions of cooccurrence:
  - in a linguistic relationship with each other (e.g., w_1 is a modifier of w_2) or
  - in the same sentence or
  - in the same document or
  - within a distance of at most k words (where k is a parameter)

Word cooccurrence in Wikipedia: Examples

- Here: cooccurrence defined as
  occurrence within $k = 10$ words of each other
- corpus = English Wikipedia
  - cooc.(rich,silver) = 186
  - cooc.(poor,silver) = 34
  - cooc.(rich,disease) = 17
  - cooc.(poor,disease) = 162
  - cooc.(rich,society) = 143
  - cooc.(poor,society) = 228

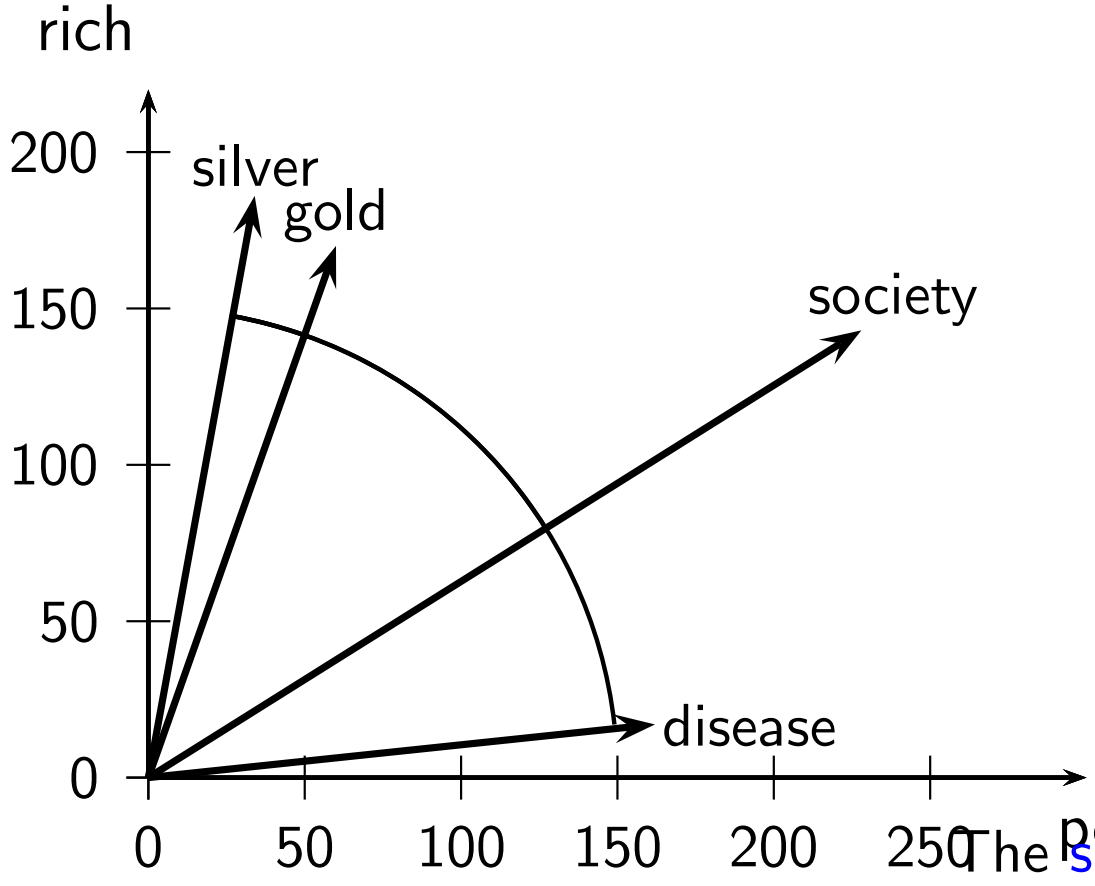
Cooccurrence counts → Count vectors



cooc.(poor,silver)=34,

cooc.(rich,silver)=186, cooc.(poor,disease)=162,
cooc.(rich,disease)=17, cooc.(poor,society)=228,
cooc.(rich,society)=143

Cooccurrence counts → Vectors → Similarity

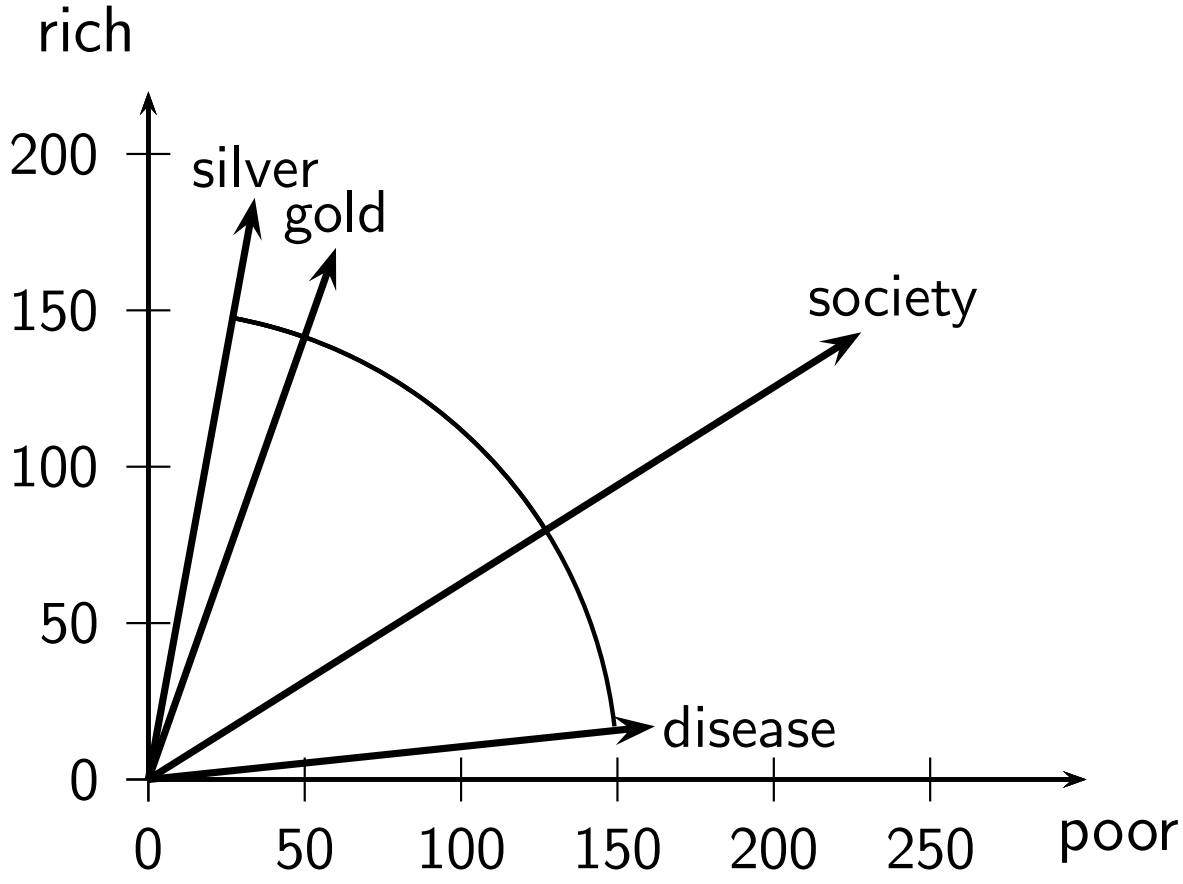


The similarity between two words is the cosine of the angle between them.
Small angle: silver and gold are similar. Medium-size angle: silver and society are not very similar. Large angle: silver and disease are even less similar.

Dimensionality of vectors

- Up to now we've only used two dimension words: rich and poor
- Now do this for a very large number of dimension words: hundreds, thousands, or even millions of dimension words.
- This is now a very high-dimensional space with a large number of vectors represented in it.
- But formally, there is no difference to a two-dimensional space with four vectors.
- Note: a word has dual role in the vector space
  - Each word is a dimension word, an axis of the space.
  - But each word is also a vector in that space.

Same formalism, but more dimensions & more vectors



Nearest neighbors of “silver” in vector space

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

Cases where distributional semantics fails

- Antonyms are judged to be similar: “disease” and “cure”.
- Ambiguity: “Cambridge”
- Non-specificity (occurs in a large variety of different contexts and has few/no specific semantic associations): “person”
- The corpus meaning is different from the meaning that comes to mind when the word is encountered without context: “umbrella”.
- Tokenization issues: “metal”

PMI: Weighting of cooccurrence counts

- PMI: pointwise mutual information
- $\text{PMI}(w_1, w_2) = \log \frac{P(w_1 w_2)}{P(w_1)P(w_2)}$
- We are replacing the raw cooccurrence count with PMI, a measure of surprise.

PMI: Weighting of cooccurrence counts

- $\mathrm{PMI}(w_1, w_2) = \log \frac{P(w_1 w_2)}{P(w_1)P(w_2)}$,
  a measure of surprise
- If w_1, w_2 independent:
  $\mathrm{PMI}(w_1, w_2) = 0$
- If w_1, w_2 perfectly correlated:
  $\mathrm{PMI}(w_1, w_2) = \log[1/P(w_2)]$
- If w_1, w_2 positively correlated:
  $\mathrm{PMI}(w_1, w_2)$ is large and positive.
- If w_1, w_2 negatively correlated:
  $\mathrm{PMI}(w_1, w_2)$ is large and negative.

PPMI

- PPMI =
  positive pointwise mutual information
- $\text{PPMI}(w_1, w_2) = \max(0, \text{PMI}(w_1, w_2))$
- More generally (with offset k):
  $\text{PPMI}(w_1, w_2) = \max(0, \text{PMI}(w_1, w_2) - k)$

Motivation for using PPMI instead of PMI

- $\text{PPMI}(w_1, w_2) = \max(0, \text{PMI}(w_1, w_2) - k)$
- Most interesting correlations of the sort we're interested in are positive. (It is very hard to find negative correlations among words that are meaningful.)
- Motivation for offset:
  Small correlations may be due to noise,
  so discard them as well.

Summary: How to build count-based distributional vectors

- Select a corpus
- Select k dimension words
- Select n focus words – these will be represented as points or vectors in the space
- Compute $k \times n$ cooccurrence matrix
- Compute (PPMI-) weighted cooccurrence matrix
- Compute similarity of any two focus words as the cosine of their vectors

Bag of words model

- We do not consider the order of words in a context.
- *John is quicker than Mary* and *Mary is quicker than John* give rise to same cooccurrence counts.
- This is called a bag of words model.
- More sophisticated models: compute dimension features based on the parse of a sentence – the feature "is object of the verb cook" would be recovered from both "John cooked the ham" and "the ham was cooked".

Summary distributional semantics

- The meaning of a word is learned from its contexts in a large corpus.
- The main analysis method of contexts is co-occurrence.
- Distributional semantics is a good model of semantic similarity.
- There is a lot more in semantics that distributional semantics is not a good model for.

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

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 $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

Embeddings

Definition

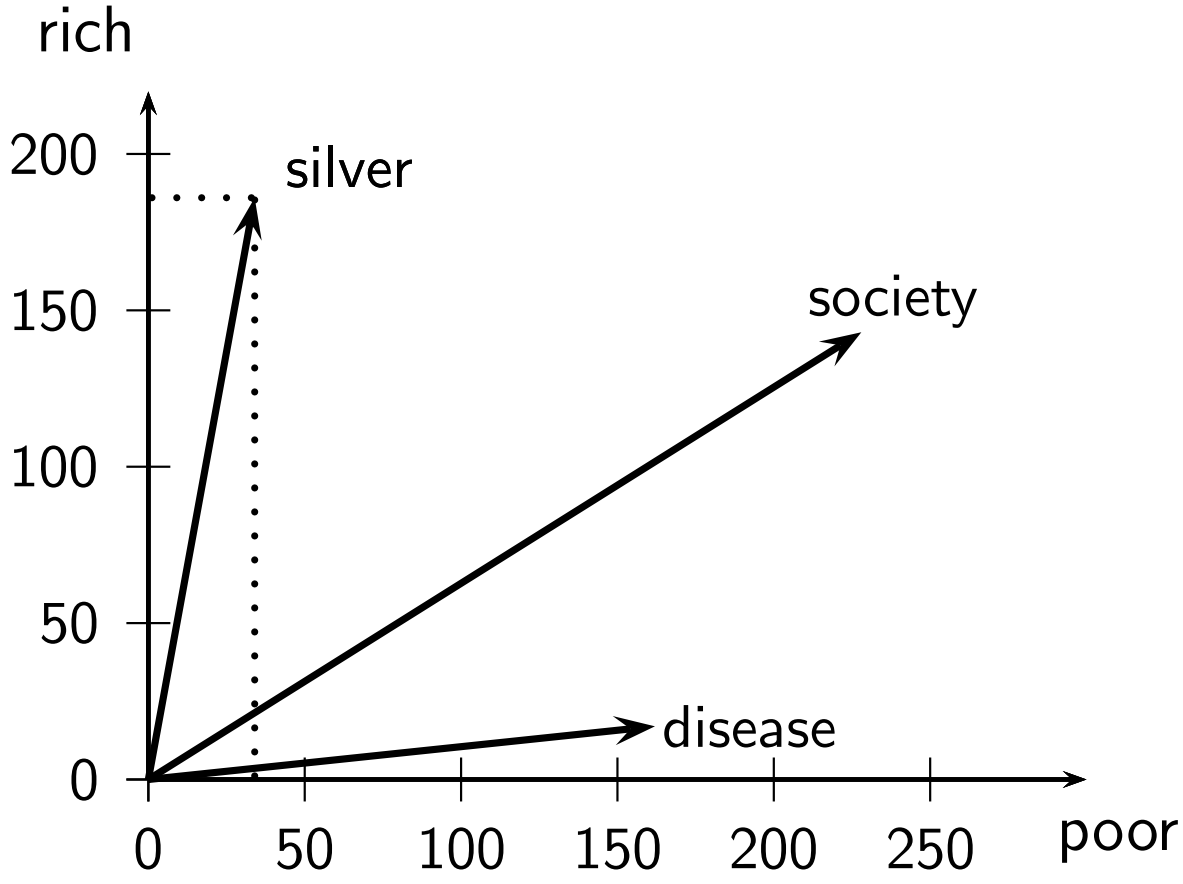
The embedding of a word w is a dense vector $\vec{v}(w) \in \mathcal{R}^k$ that represents semantic and other properties of w. Typical values are $50 \leq k \leq 1000$.

- It appears there is little difference to count vectors: Both embeddings and count vectors are representations of words, primarily semantic, but also capturing other properties.
- Embeddings have much lower dimensionality than count vectors.
- Count vectors are sparse (most entries are 0), embeddings dense (almost never happens that an entry is 0).

Example of an embedding vector

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)

Count vectors: Why are embeddings better?



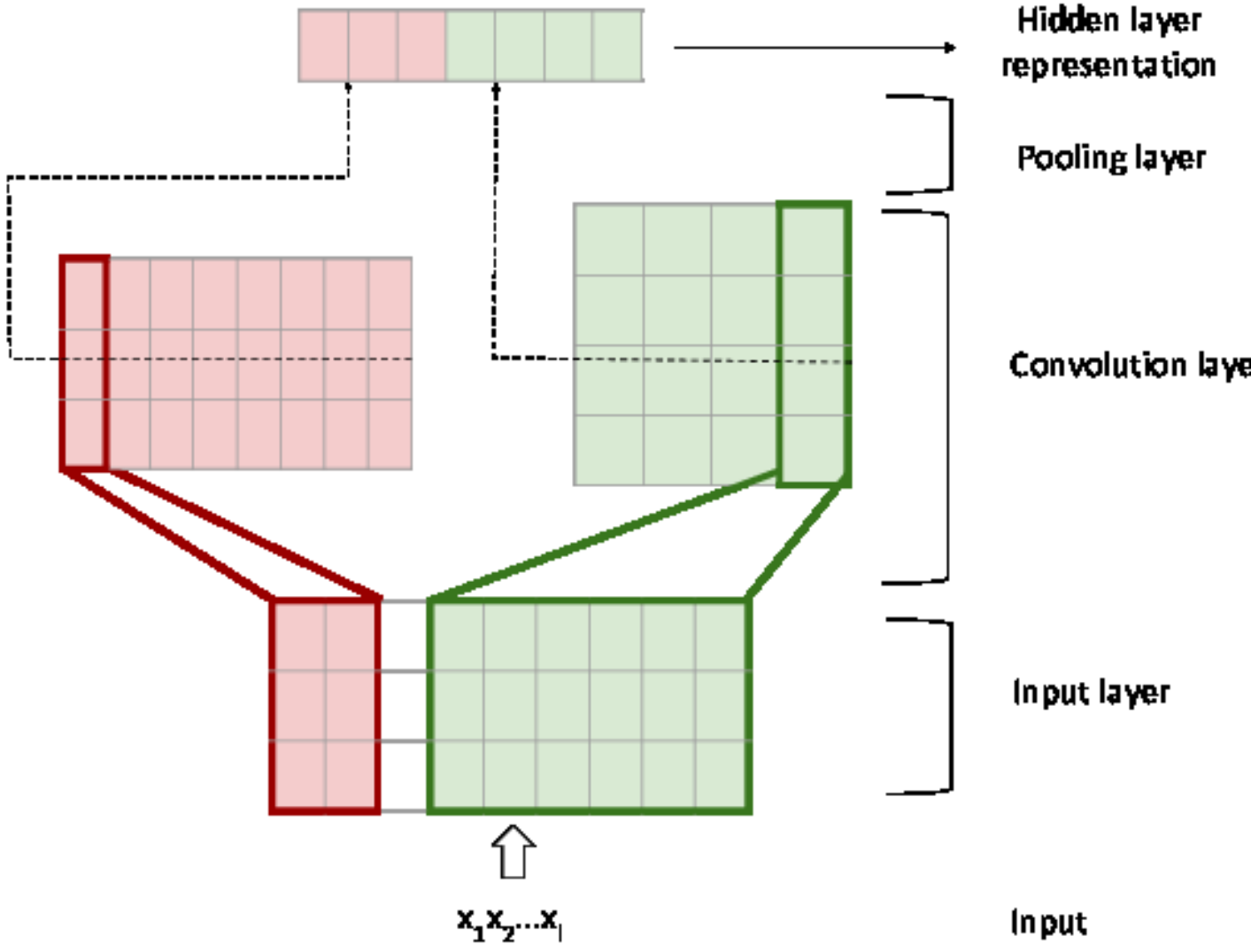
Count vectors vs. embeddings

- Count 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 $\rightarrow$ slow training

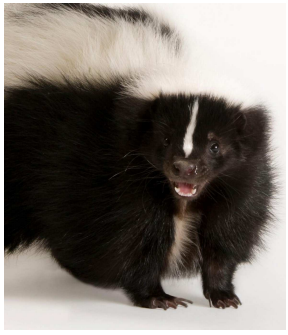
The time to train a neural network is roughly linear in the dimensionality of word vectors.



Count vectors: Example for non-effectiveness

- Example: polarity classification
- Cooccurrence with "bad" indicates negative polarity.
- But cooccurrence counts are often random and noisy and a negative word may not have occurred with "bad".
- Possible result:
  Incorrect classification based on count vectors

Ineffectiveness of count vectors: Polarity



$$\begin{pmatrix} \text{high:0} \\ \text{cold:2} \\ \text{tree:2} \\ \text{good:0} \\ \text{white:1} \\ \text{bad:0} \\ \text{sweet:0} \\ \text{smelly:5} \end{pmatrix} \rightarrow$$

positive

neutral

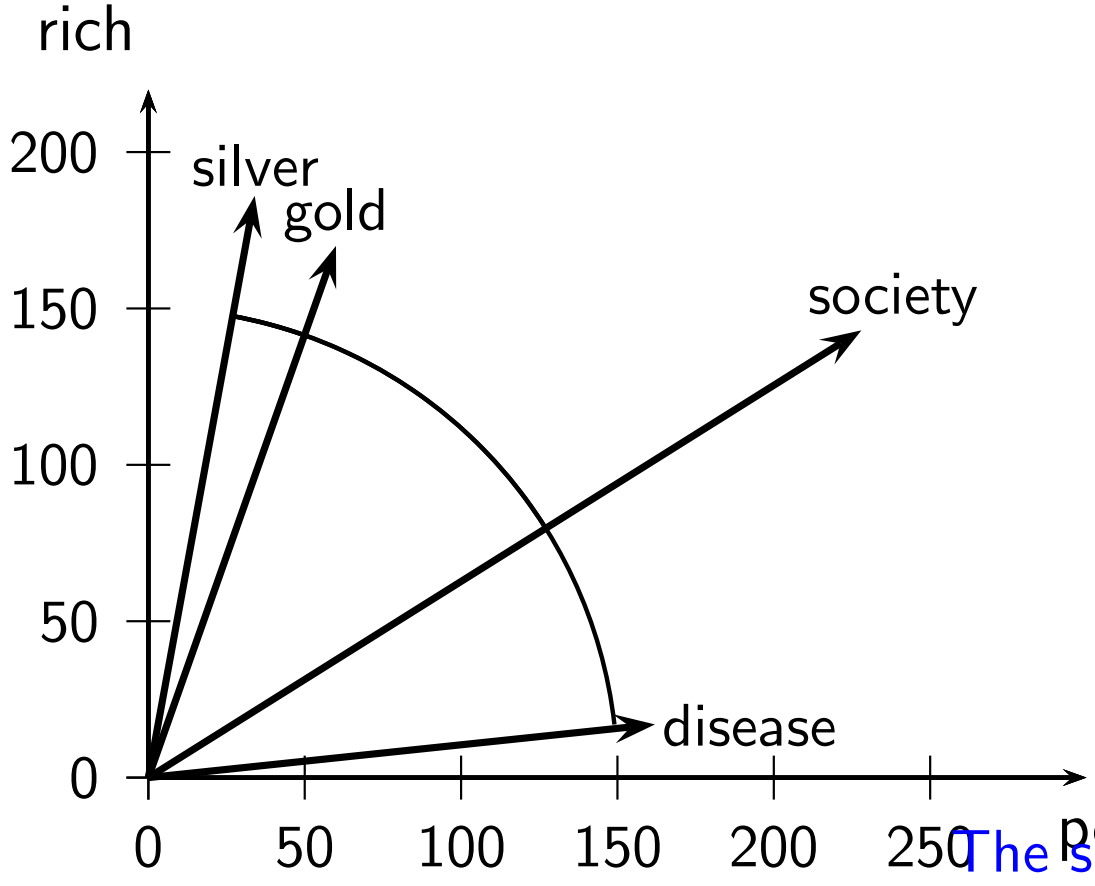
negative

$\vec{v}(\text{skunk})$

Embedding learning algorithms

- word2vec skipgram
  - Language model optimized by gradient descent
  - Best known approach
- GloVe
  - Factorization of cooccurrence matrix
  - Least squares objective optimized by gradient descent
- fastText
  - Character-level model
  - Next section
- Singular Value Decomposition (SVD)
  - Also called Latent Semantic Indexing (LSI)
  - Factorization of cooccurrence matrix
  - Least squares objective optimized by power method

Basis for count vector space: Cooccurrence



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 even less similar.

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

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

Cooccurrence count (CC) matrix

		w_2				
		rich	poor	silver	society	disease
w_1	rich	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$
	poor	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$
	silver	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$
	society	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$
	disease	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$	$CC(w_1, w_2)$

PPMI matrix C

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

		w_2				
		rich	poor	silver	society	disease
w_1	rich	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)
	poor	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)
	silver	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)
	society	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)
	disease	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)	PPMI(w_1, w_2)

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 and more compact word representations out of C' (compared to C).

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

C	w_1	w_2	w_3	w_4	w_5	w_6
rich	1	0	1	0	0	0
poor	0	1	0	0	0	0
silver	1	1	0	0	0	0
society	1	0	0	1	1	0
disease	0	0	0	1	0	1

$=$

U	1	2	3	4	5
rich	−0.44	−0.30	0.57	0.58	0.25
poor	−0.13	−0.33	−0.59	0.00	0.73
silver	−0.48	−0.51	−0.37	0.00	−0.61
society	−0.70	0.35	0.15	−0.58	0.16
disease	−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
4	0.00	0.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.39

$\times$

V^T	w_1	w_2	w_3	w_4	w_5	w_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

SVD is decomposition of C into a representation of the input words, a representation of the context and a representation of the importance of the "semantic" dimensions.

embeddings = left singular vectors

U	1	2	3	4	5
rich	−0.44	−0.30	0.00	0.00	0.00
poor	−0.13	−0.33	0.00	0.00	0.00
silver	−0.48	−0.51	0.00	0.00	0.00
society	−0.70	0.35	0.00	0.00	0.00
disease	−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	w_1	w_2	w_3	w_4	w_5	w_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

SVD: Summary

- We've decomposed the cooccurrence matrix C into a product of three matrices: $U\Sigma V^T$.
- The input word matrix U – consists of one (row) vector for each word
- The context word matrix V^T – consists of one (column) vector for each context word
- The singular value matrix Σ – diagonal matrix with singular values, reflecting importance of each dimension
- We only keep first k dimensions and set the others to zero.

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

How to learn embeddings via SVD

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

SVD embeddings vs. word2vec skipgram

- Levy&Goldberg try to prove:
  word2vec skipgram equivalent to SVD of PPMI matrix
- Important because it links two important bodies of research:
  distributional semantics and embeddings

word2vec skipgram equivalent to SVD of PPMI matrix

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.

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

skipgram objective

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} -\log p(w_c | w_t)$$

T	length of the training corpus in tokens
$\mathcal{C}_t$	words surrounding word w_t

Probability of a context word: softmax?

$$p(w_c|w_t) = \frac{\exp(s(w_t, w_c))}{\sum_{j=1}^{W} \exp(s(w_j, w_c))}$$

$s(w_t, w_c)$ scoring function that maps word pair to $\mathcal{R}$

Problems: too expensive

Instead of softmax:

Negative sampling and binary logistic loss

$$\log(1 + \exp(-s(w_t, w_c))) + \sum_{n \in \mathcal{N}_{t,c}} \log(1 + \exp(s(w_t, w_n)))$$

$$\ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, w_n))$$

$\mathcal{N}_{t,c}$ set of negative examples sampled from the vocabulary

$\ell(x)$ $\log(1 + \exp(-x))$ (logistic loss)

Binary logistic loss for corpus

$$\sum_{t=1}^{T} \sum_{c \in \mathcal{C}_t} \left[\ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, w_n)) \right]$$

$\ell(x) \quad \log(1 + \exp(-x))$

Scoring function

$$s(w_t, w_c) = \mathbf{u}_{w_t}^{\mathsf{T}} \mathbf{v}_{w_c}$$

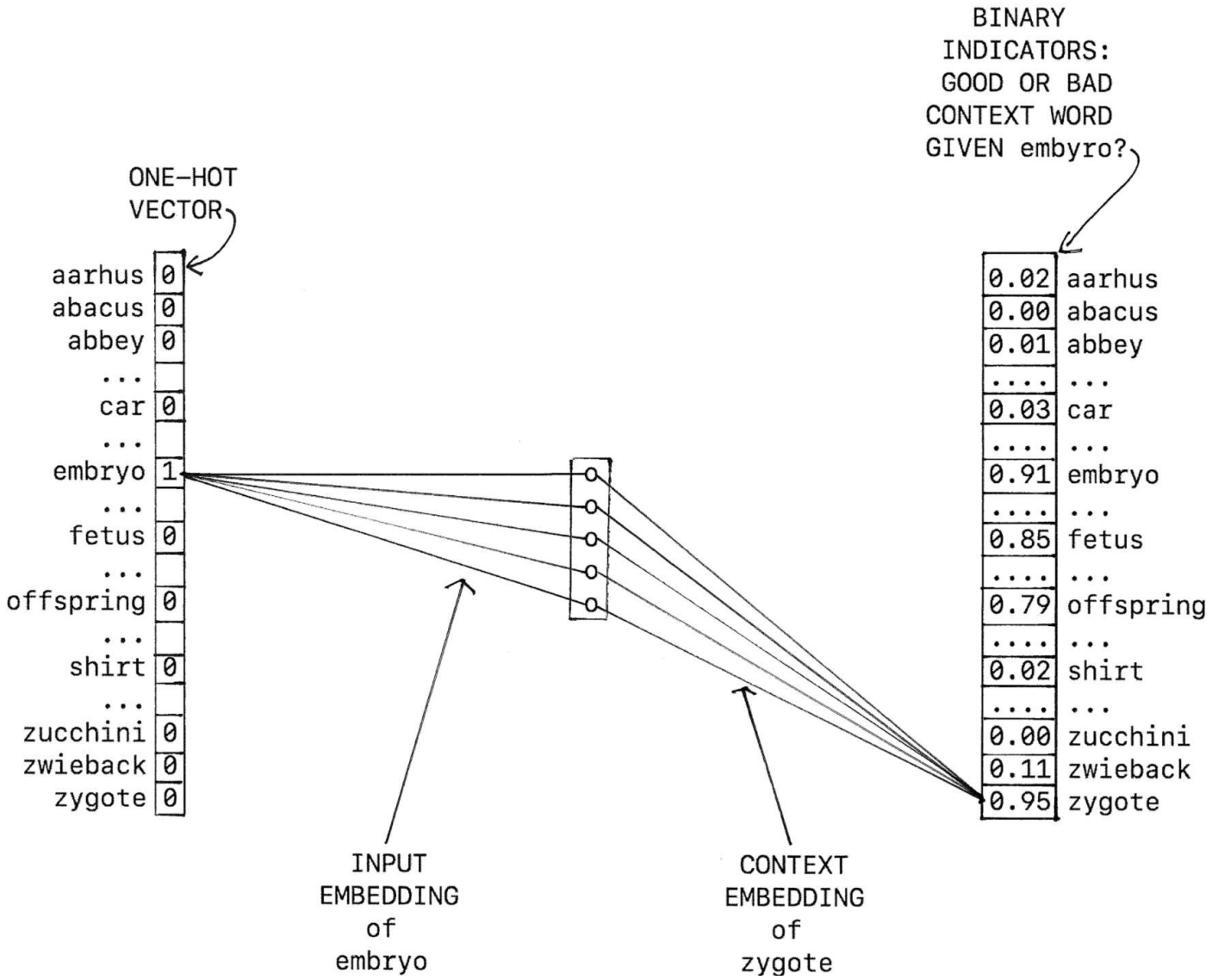
$\mathbf{u}_{w_t}$ the input vector of w_t

$\mathbf{v}_{w_c}$ the output vector (or context vector) of w_c

Input vs. context embeddings

- Note that each word has two embeddings:
  an input embedding and a context embedding
- We generally only use the input embedding.

Is word2vec a neural network?



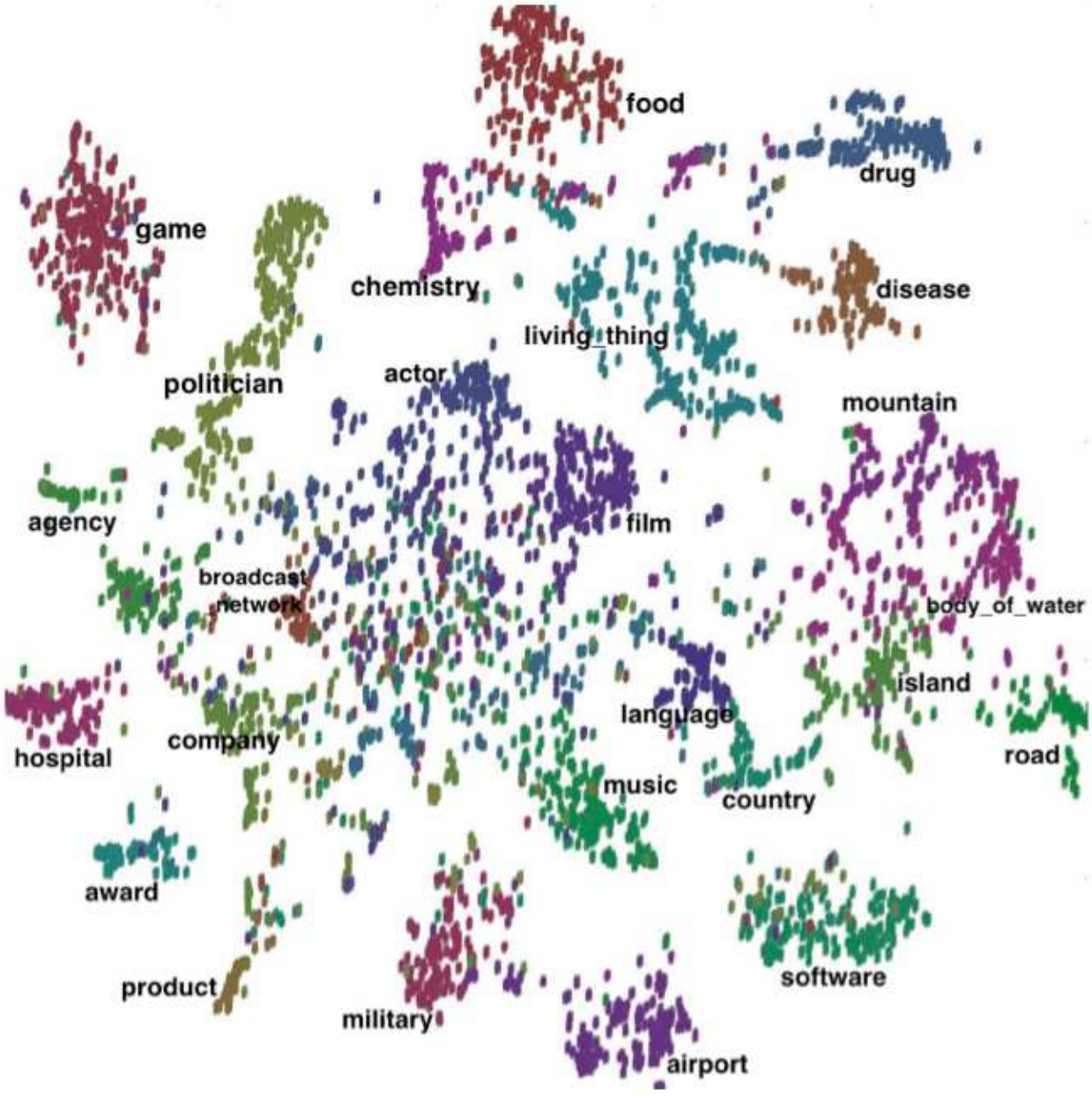
word2vec: Summary

- Language modeling objective
- Trained by gradient descent
- Negative sampling to make training efficient
- "Compatibility" of focus word and context word is scored as dot product of two vectors, the input embedding and the context embedding.
- We usually use the input embedding.
- Closely related to count vectors: we can think of embeddings as dimensionality-reduced count vectors.

Visualization

- How to understand / analyze embeddings?
- Frequently used: two-dimensional projections
- Methods / software
  - Traditional:
    multidimensional scaling, PCA
  - t-SNE
    `https://lvdmaaten.github.io/tsne/`
  - gensim
    `https://radimrehurek.com/gensim/`
  - Pretty much all methods are implemented in R:
    `https://www.r-project.org`
- Important: The two dimensions are not interpretable.

2D projection of entity embeddings



GoogleNews embeddings

- `https://code.google.com/archive/p/word2vec`
- Embedding basis for `http://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

GloVe embeddings

https://nlp.stanford.edu/projects/glove

Resources

- See “references.pdf” document
- `https://bitbucket.org/yoavgo/word2vecf/src/default/`

Questions?