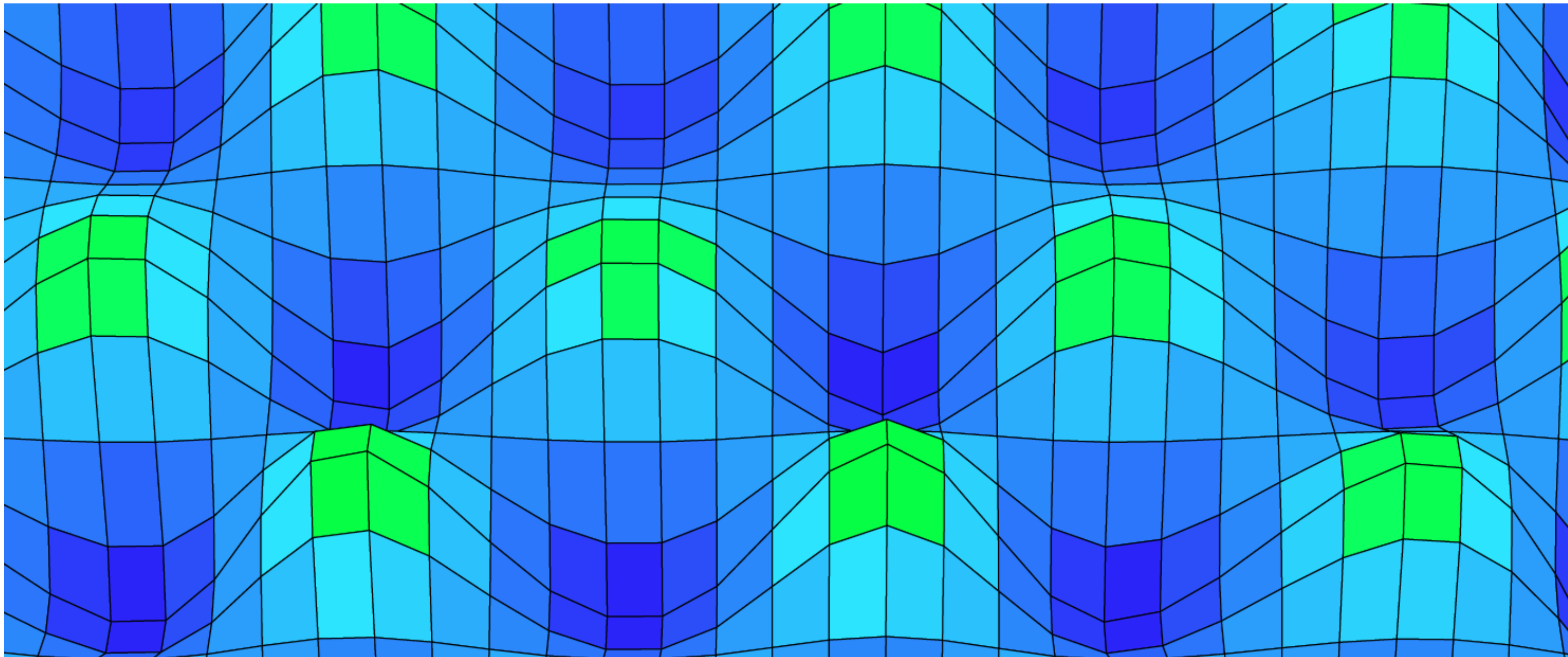
EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Philosophische Fakultät
Seminar für Sprachwissenschaft



Cognitive models of language processing: vector semantics

Peter Hendrix

Credits

Speech and Language Processing:
An Introduction to Natural Language Processing

Jurafsky & Martin (2009)

Definition of meaning

What is meaning?

Dictionary definitions

- man: “an adult human male”
- woman: “an adult human female”
- boy: “a male child or youth”
- girl: “a female child or youth”

Semantic features

- man: [+HUMAN], [+MALE], [+ADULT]
- woman: [+HUMAN], [-MALE], [+ADULT]
- boy: [+HUMAN], [+MALE], [-ADULT]
- girl: [+HUMAN], [-MALE], [-ADULT]

Semantic features

- Problems:
  - features are discrete:
    - hot: [+WARM]
    - cold: [-WARM]
    - lukewarm: [?WARM]
    - human: [?WARM]
  - potentially infinite set of features
  - selection of features is subjective and labour-intensive

Distributional semantics

"You shall know a word by the company it keeps"

John Rupert Firth

Distributional semantics

- What does lamap mean?
  - Where did you buy this lamap?
  - Sit on the lamap and make yourself comfortable.
  - After we moved the lamap, there was a round faded area on the floor.

Nomenclature

distributional semantics

=

vector semantics

=

vector-space semantics

Why “vector semantics”?

- Word meanings are represented as a vector of numbers
- The numbers are based on co-occurrence frequencies

Vector semantics: steps

- Steps:

  1) Calculate co-occurrence matrix
  2) Apply weighting scheme
  3) Reduce dimensionality
  4) Calculate similarity between semantic vectors

Vector semantics: steps

- Steps:
  1) Calculate co-occurrence matrix
  2) Apply weighting scheme
  3) Reduce dimensionality
  4) Calculate similarity between semantic vectors

Co-occurrence matrix

- Define co-occurrence of words with contexts
- Contexts can be:
  - documents $\rightarrow$ term-document matrix
  - $\ldots$
  - words $\rightarrow$ term-term matrix
- Retrieve co-occurrence frequency for all word-context pairs

Types of co-occurrence matrices

- Term-document matrix:
  - topic retrieval
  - search results optimization
- Term-term matrix:
  - semantic similarities between words
  - semantic categorization
  - sentiment analysis
  - ...

Term-document matrix

	text1	text2	text3	text4	text5	...
table	0	3	1	3	3	...
house	0	5	1	4	1	...
cat	0	0	7	2	0	...
banana	5	0	0	1	2	...
apple	4	1	1	4	2	...
...	...	...	...	...	...	...

Term-term matrix

- Define a window size *n* (e.g.; 5)
- Context words are all words within *n* words of the target word
- Calculate a co-occurrence matrix

Term-term matrix

. . .

“. . . the room there was a table, nicely dressed with a new . . . ”

“. . . he quickly had breakfast: a banana, a glass of juice and . . . ”

“. . . she spotted the dog, the cat ran into the room and”

. . .

Term-term matrix

	room	breakfast	school	painting	party	...
table	1	1	2	5	4	...
house	3	0	1	2	2	...
cat	1	0	0	0	1	...
banana	0	4	0	2	1	...
apple	0	3	1	4	0	...
...	...	...	...	...	...	...

Term-term matrix

- The window size *n* is a free parameter
- Window size determines the type of relations captured:
  - smaller window: more syntactic
  - larger window: more semantic
- Optimal window size depends on the application

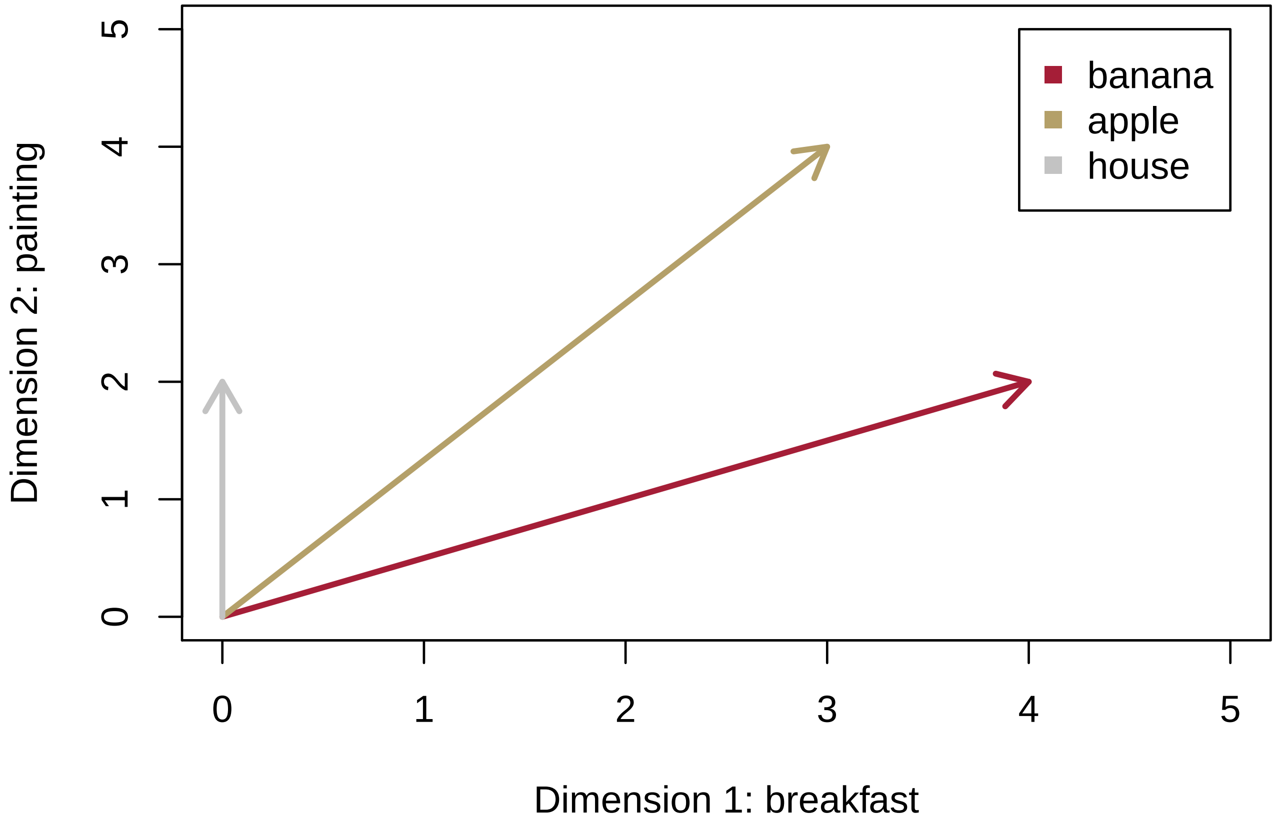
Semantic vectors

	room	breakfast	school	painting	party	...
table	1	1	2	5	4	...
house	3	0	1	2	2	...
cat	1	0	0	0	1	...
banana	0	4	0	2	1	...
apple	0	3	1	4	0	...
...	...	...	...	...	...	...

Semantic vectors: geometry

	room	breakfast	school	painting	party	...
table	1	1	2	5	4	...
house	3	0	1	2	2	...
cat	1	0	0	0	1	...
banana	0	4	0	2	1	...
apple	0	3	1	4	0	...
...	...	...	...	...	...	...

Semantic vectors: geometry



Vector semantics: steps

- Steps:
  1) Calculate co-occurrence matrix
  2) Apply weighting scheme
  3) Reduce dimensionality
  4) Calculate similarity between semantic vectors

Weighting

- Raw co-occurrence counts are suboptimal
- Adjust counts for frequency of terms in isolation
- Most common weighting scheme: point-wise mutual information (PMI)

PMI

- Compare observed frequency with expected frequency:

$$\mathrm{PMI} = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- Positive value: co-occurrence frequency higher than expected by chance

- Negative value: co-occurrence frequency lower than expected by chance

PPMI

- Negative values of PMI can only be established reliably with massive corpora
- Solution: set negative values to zero
- Positive point-wise mutual information:

$$\text{PPMI} = \max\left(0, \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}\right)$$

PPMI

	room	breakfast	school	painting	party	$\sum$
table	1	1	2	5	4	13
house	3	0	1	2	2	8
cat	1	0	0	0	1	2
banana	0	4	0	2	1	7
apple	0	3	1	4	0	8
$\sum$	5	8	4	13	8	38

$$\mathrm{PPMI}_{banana,breakfast} = \max\left(0, \log_2 \frac{\frac{4}{38}}{\frac{7}{38} * \frac{8}{38}}\right) = 1.44$$

PPMI

	room	breakfast	school	painting	party
table	0	0	0.55	0.17	0.55
house	1.51	0	0.25	0	0.25
cat	1.92	0	0	0	1.25
banana	0	1.44	0	0	0
apple	0	0.83	0.25	0.55	0

PPMI

	room	breakfast	school	painting	party
table	0	0	0.55	0.17	0.55
house	1.51	0	0.25	0	0.25
cat	1.92	0	0	0	1.25
banana	0	1.44	0	0	0
apple	0	0.83	0.25	0.55	0

PPMI

- PPMI is biased towards low frequency words
- Low frequency words tend to have higher PPMI values
- Solution: Laplace smoothing
- Add a low number to all frequency counts

Laplace smoothing

	room	breakfast	school	painting	party
table	1	1	2	5	4
house	3	0	1	2	2
cat	1	0	0	0	1
banana	0	4	0	2	1
apple	0	3	1	4	0

Laplace smoothing

	room	breakfast	school	painting	party
table	2	2	3	6	5
house	4	1	2	3	3
cat	2	1	1	1	2
banana	1	5	1	3	2
apple	1	4	2	5	1

PPMI (Laplace smoothing)

	room	breakfast	school	painting	party
table	0	0	0.22	0.22	0.43
house	0.95	0	0.11	0	0.16
cat	0.85	0	0	0	0.47
banana	0	1.01	0	0	0
apple	0	0.58	0.11	0.43	0

Weighting schemes

- PPMI is the most popular weighting schemes for term-term matrices
- Alternative:

$$\text{t-score}_{w_1 w_2} = \frac{P(w_1, w_2) - P(w_1)P(w_2)}{\sqrt{P(w_1)P(w_2)}}$$

- For term-document matrices:

$$\text{tf-idf}_{w_1 w_2} = \text{tf}_{w_1 w_2} * \log\left(\frac{N}{d_{w_1}}\right)$$

where N is the total number of documents

Vector semantics: steps

- Steps:
  1) Calculate co-occurrence matrix
  2) Apply weighting scheme
  3) Reduce dimensionality
  4) Calculate similarity between semantic vectors

Dimension reduction

- Semantic vectors from co-occurrence matrices are:
    - long
    - sparse
- Use dimension reduction techniques to reduce the length of semantic vectors
- Advantages:
    - computationally efficient
    - less overfitting
    - capture latent semantic structure

Latent semantic structure

- Higher order co-occurrences
- Example:
  - Dolphins are intelligent.
  - Whales are smart animals.
- The columns for intelligent and smart are correlated

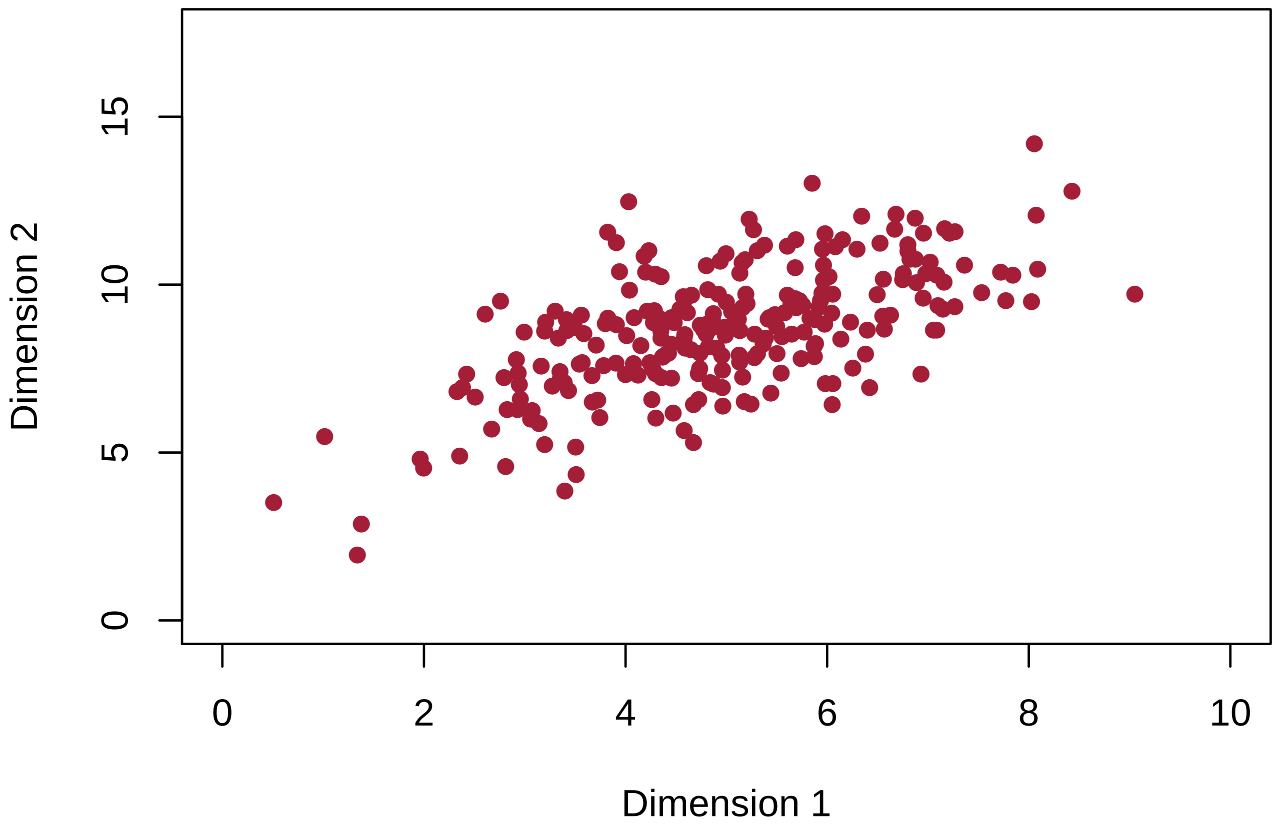
Latent semantic structure

- Can we re-organize the data such that:
  1) the information in the co-occurrence matrix is retained
  2) columns are no longer correlated
  3) the number of columns is reduced
- Use single value decomposition (SVD)
- Applicable to both term-document and term-term matrices

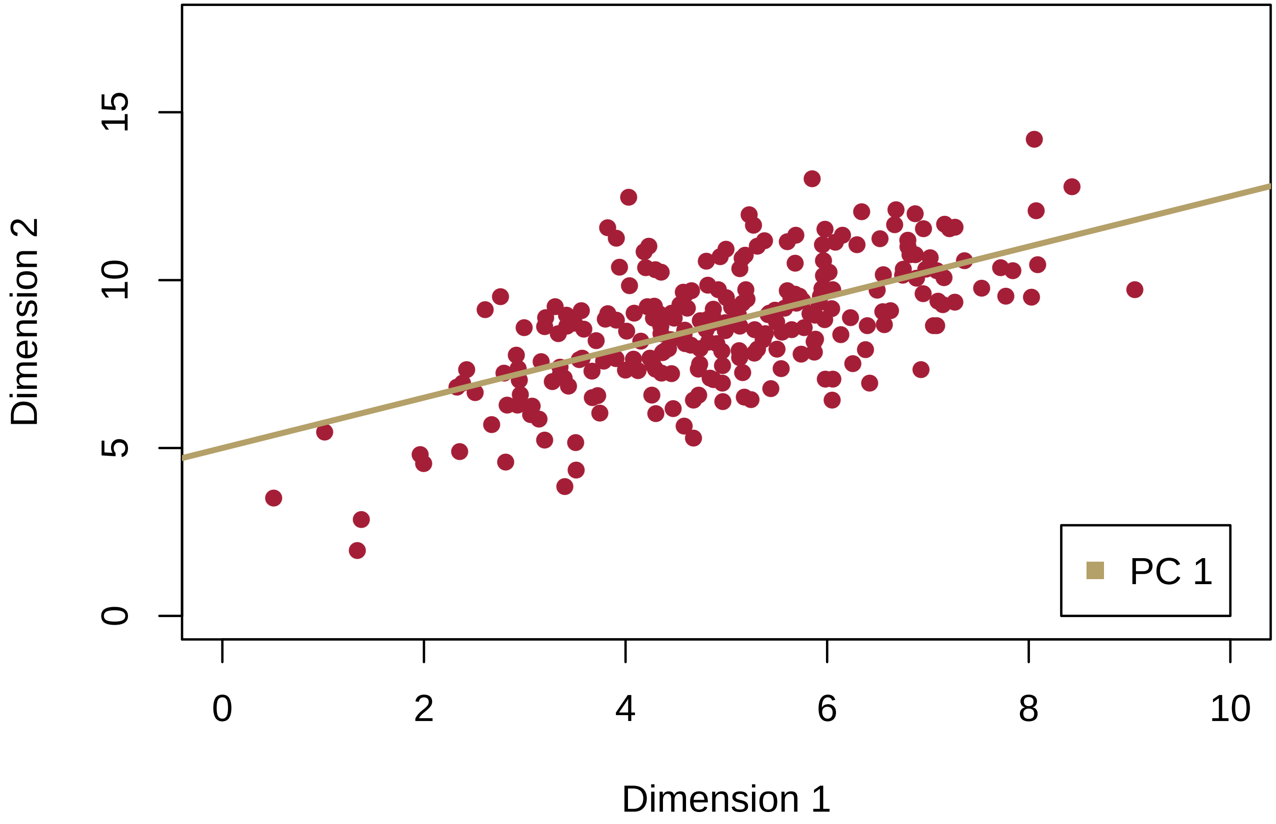
SVD

- Rotate the original n-dimensional data space:
  - sequentially find dimensions with the greatest variance in the original data
  - restriction: dimensions have to be orthogonal to all previous dimensions
- Dimensions in the rotated space are called principal components

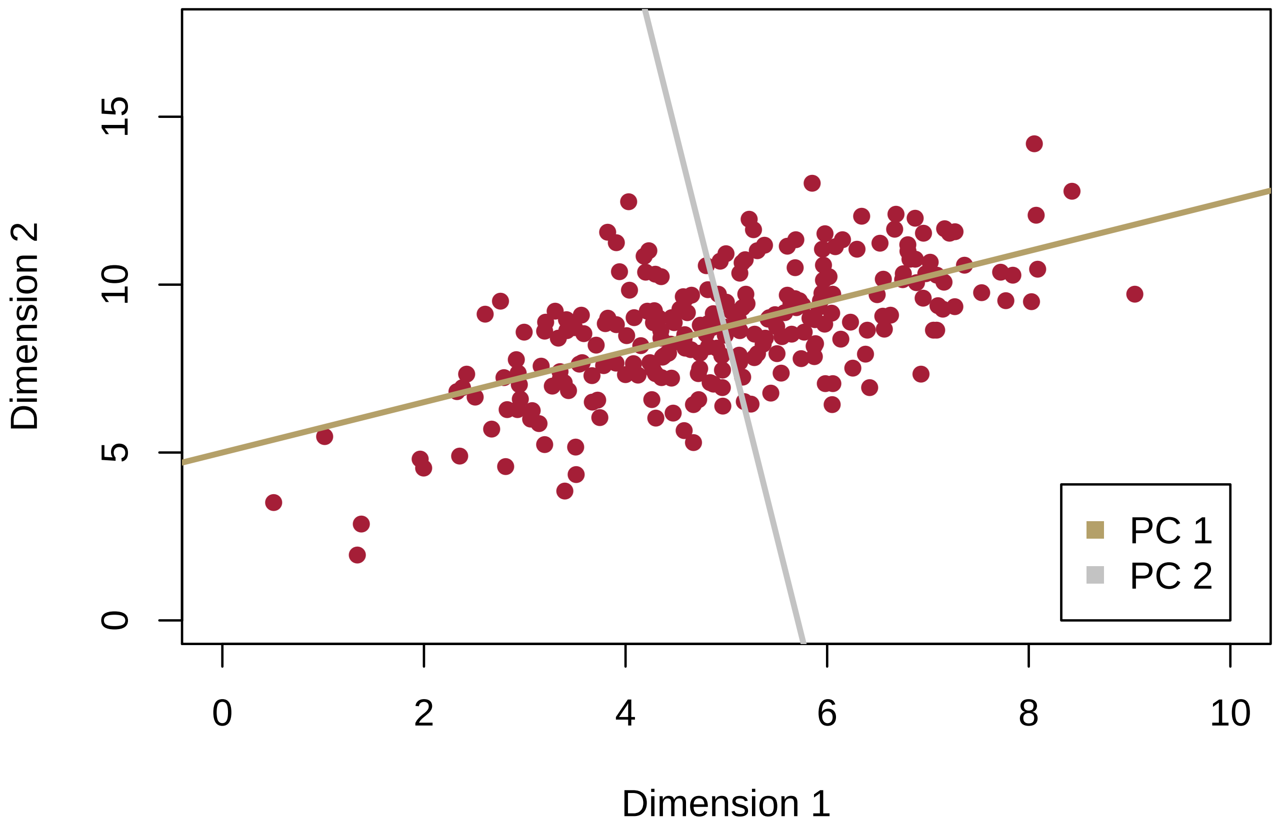
SVD



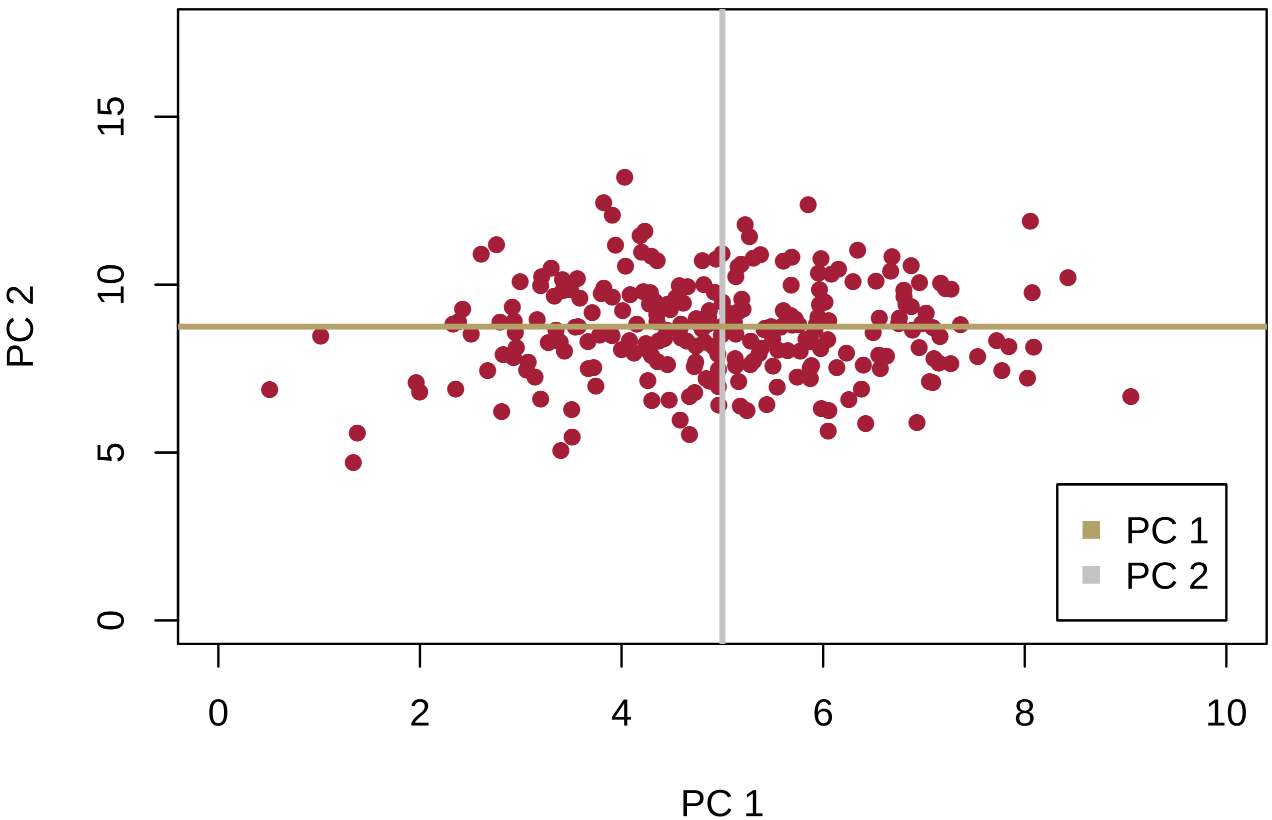
SVD



SVD



SVD



SVD

- Later dimensions explain little variance in the data
- Reduce to the rotated semantic space to *n* dimensions
- Typical values for *n*: 100 to 5000

Vector semantics: steps

- Steps:
  1) Calculate co-occurrence matrix
  2) Apply weighting scheme
  3) Reduce dimensionality
  4) Calculate similarity between semantic vectors

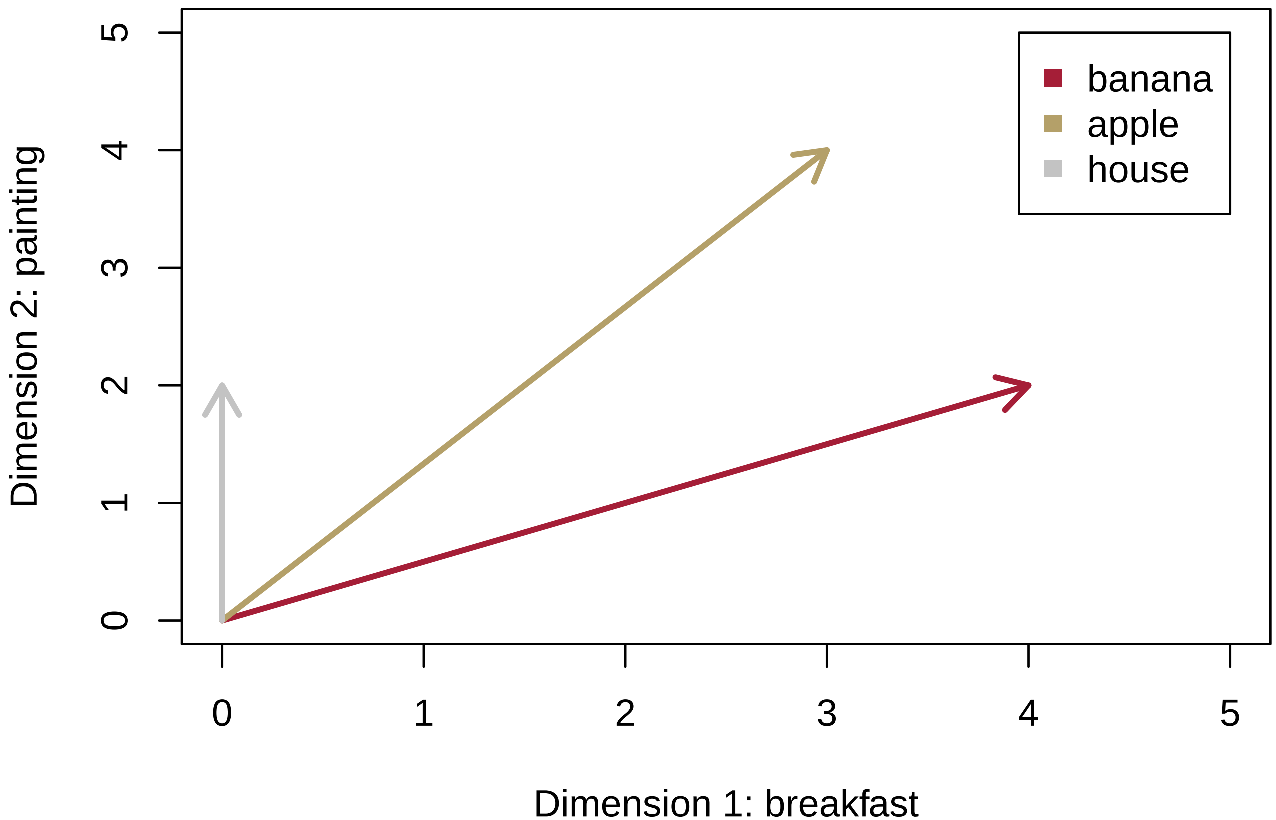
Similarity between semantic vectors

- Semantic vectors are:
  - rows in the (processed) co-occurrence matrix
  - representations of word meanings
  - meaningful only in the context of other semantic vectors from the same semantic space
- We need a measure of the similarity of semantic vectors

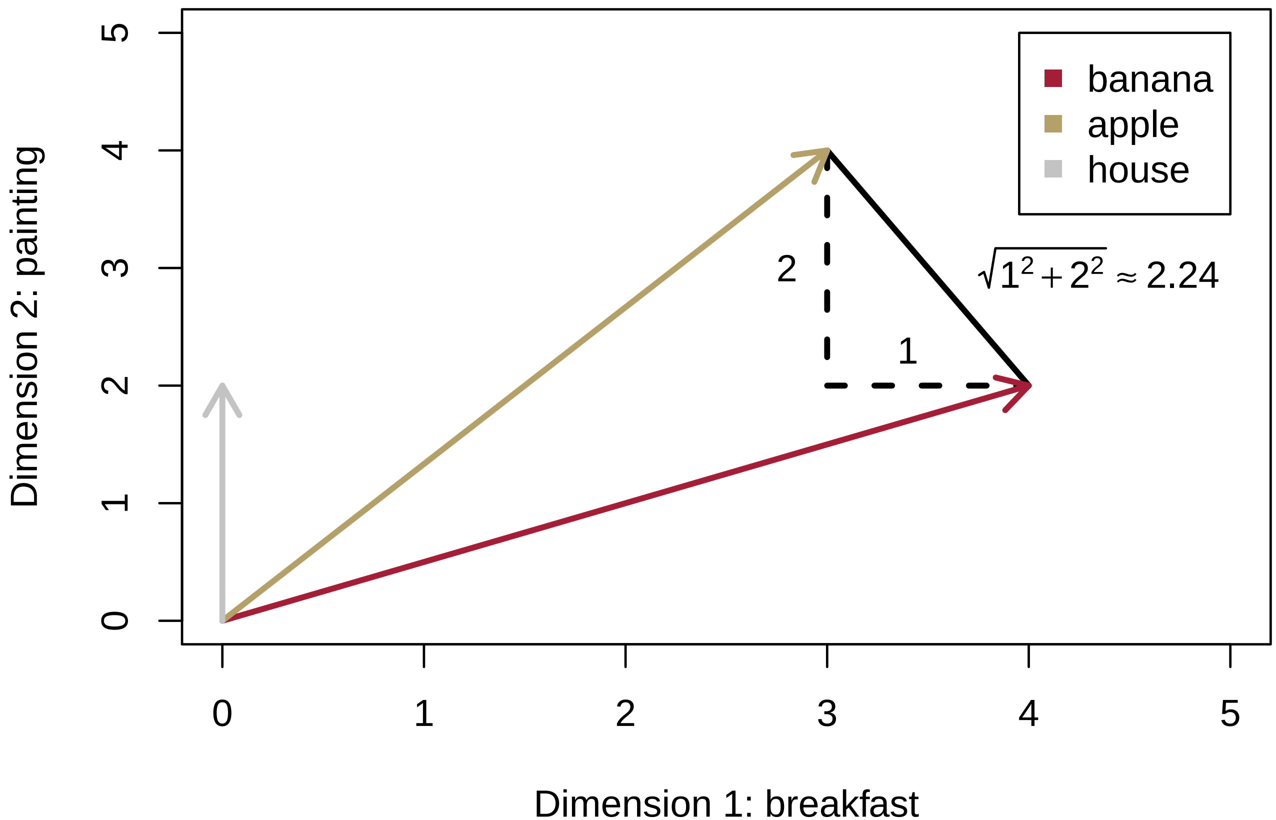
Semantic similarity metric

- Semantic similarity is inversely proportional to the distance between semantic vectors
- Distance metrics:
    - Euclidean distance
    - Cosine similarity

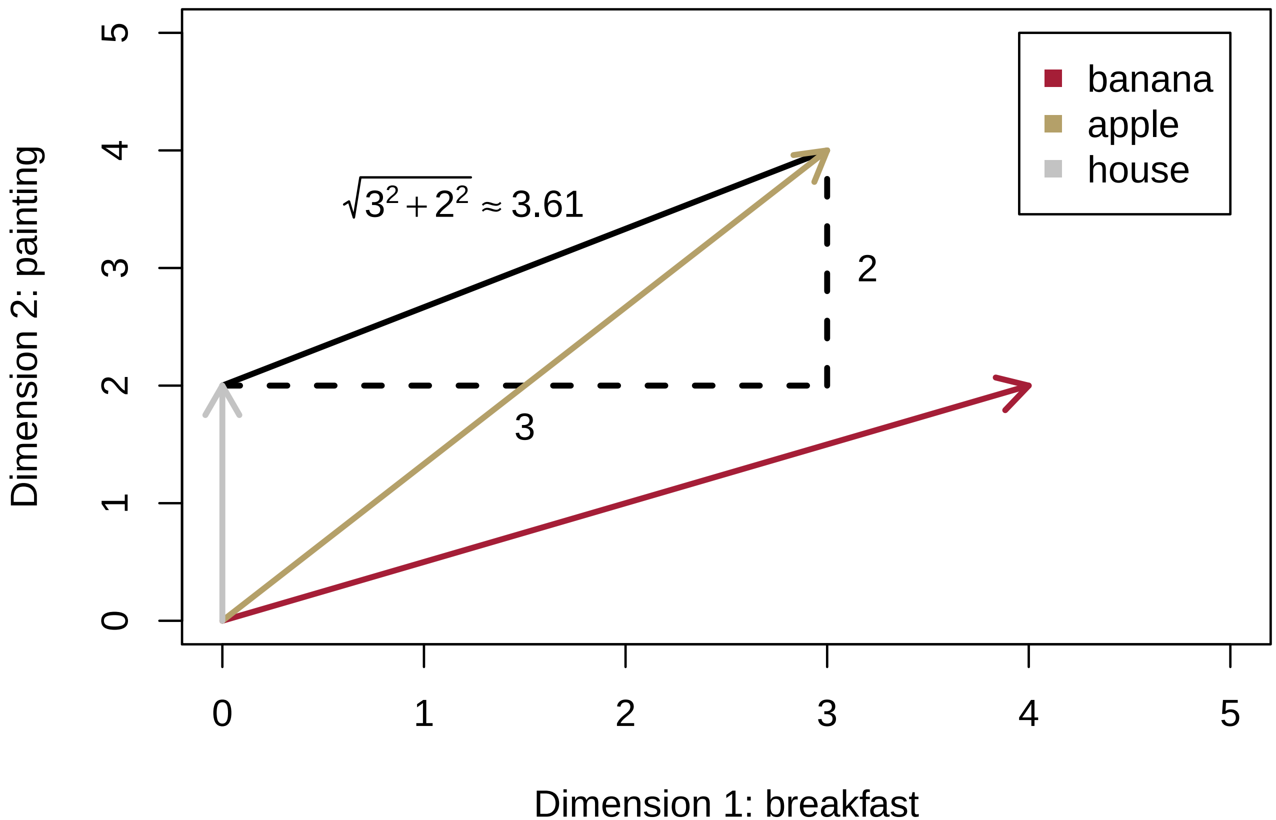
Euclidean distance



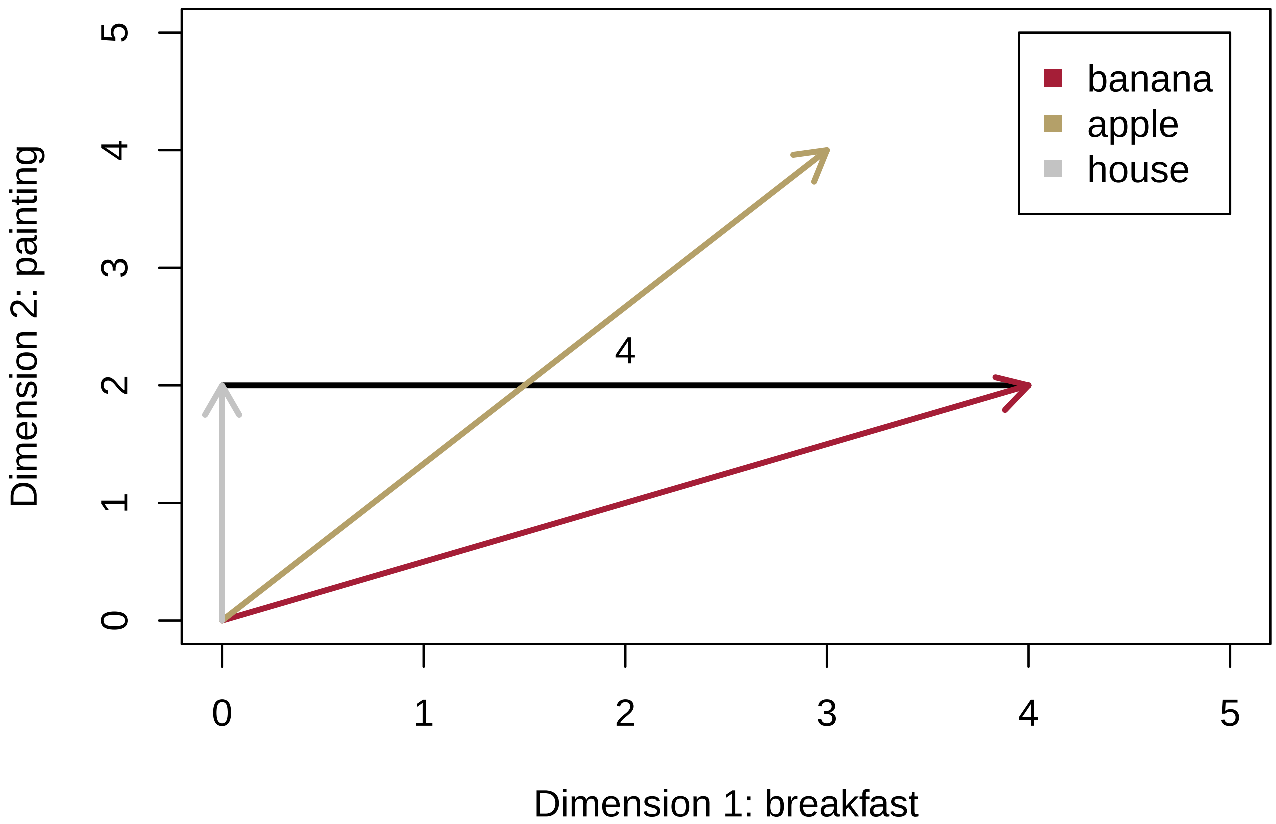
Euclidean distance



Euclidean distance



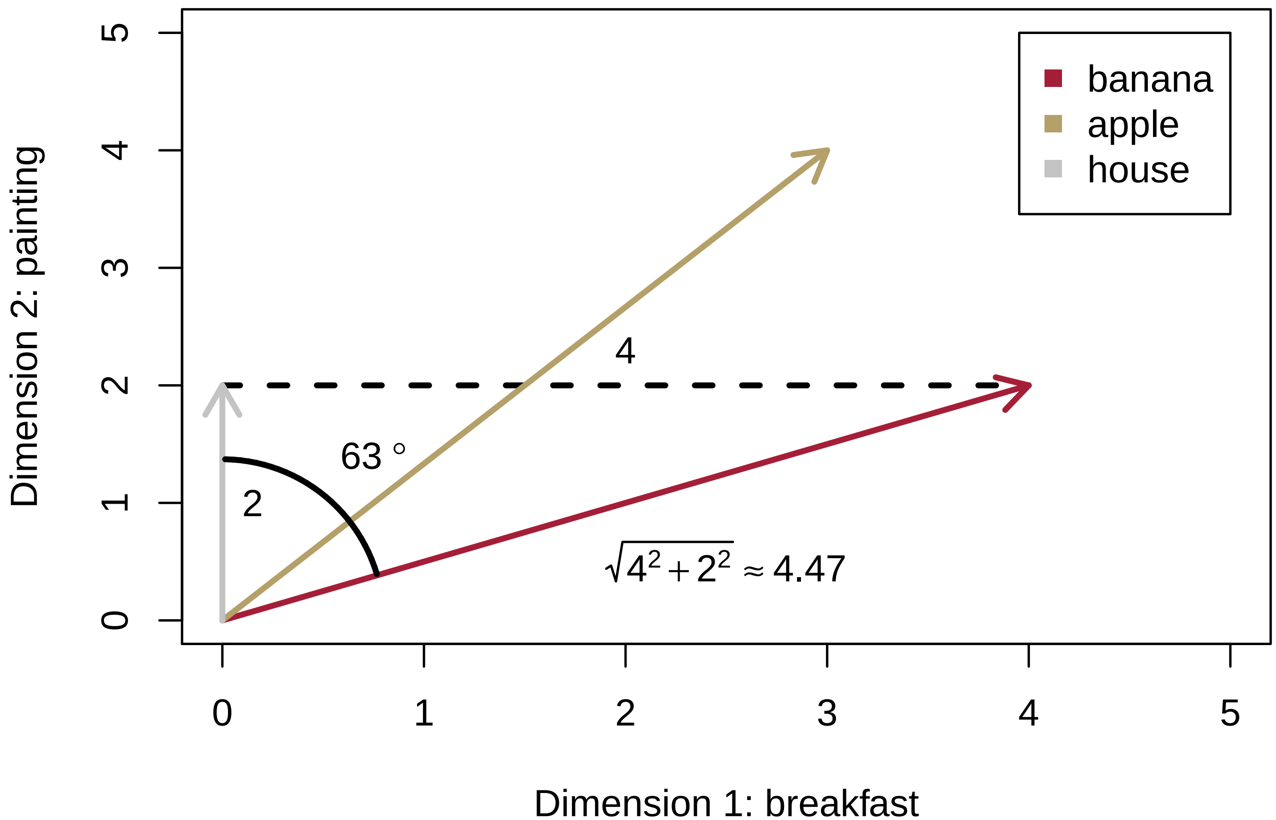
Euclidean distance



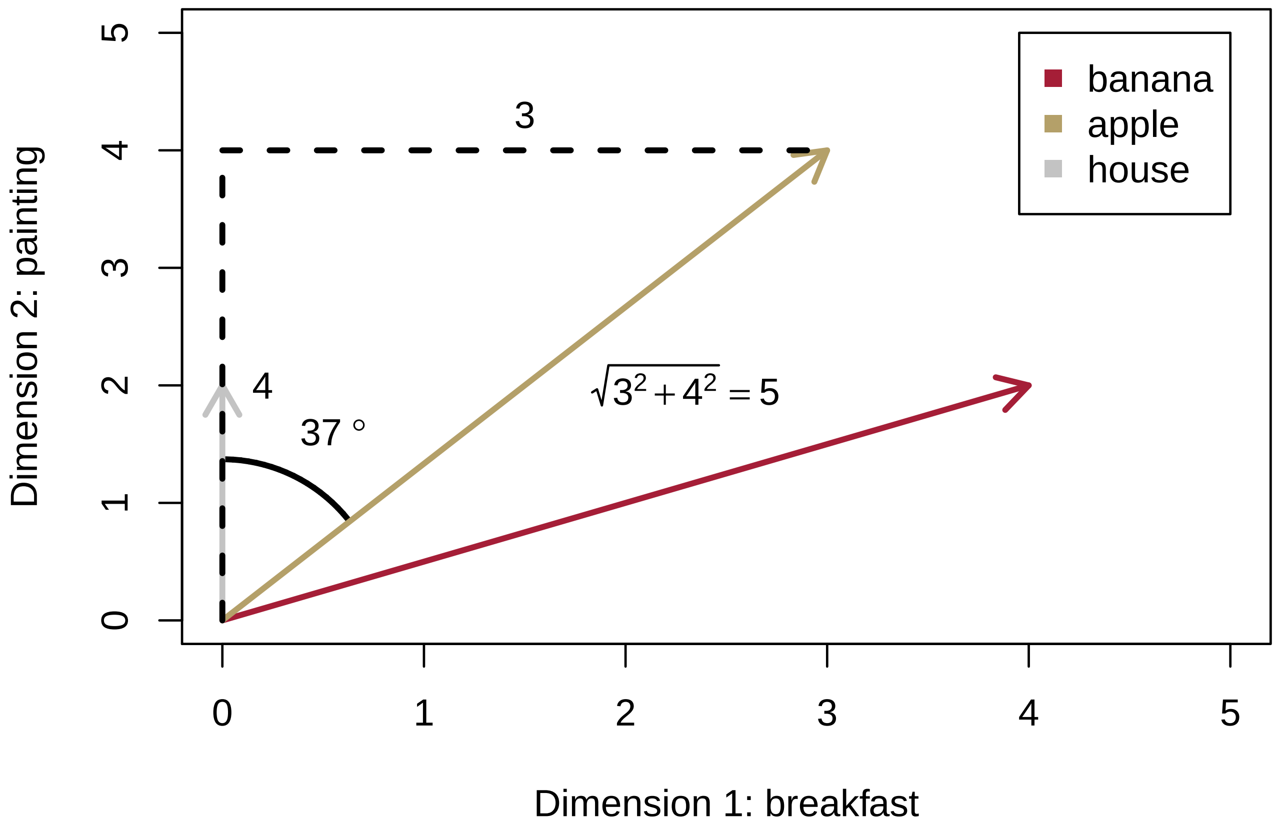
Euclidean distance

- Euclidean distance between semantic vectors:
  - banana - apple ≈ 2.24
  - banana - house ≈ 4.47
  - apple - house ≈ 3.61
- Euclidean distance between semantic vectors is inversely proportional to semantic similarity
- Euclidean distance is sensitive to vector length

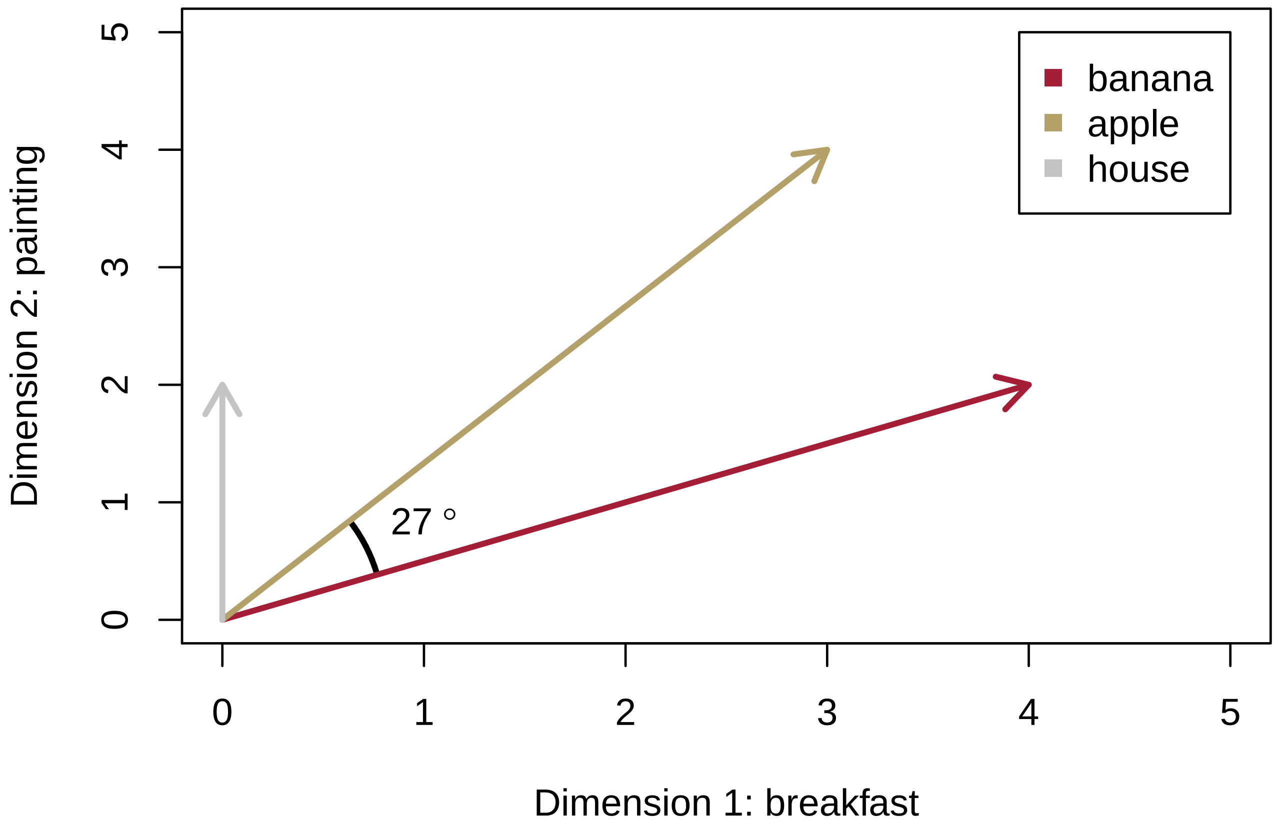
Cosine distance



Cosine distance



Semantic similarity: cosine distance



Cosine distance

- Angles between semantic vectors:
  - banana - apple $\approx$ 26.56
  - banana - house $\approx$ 63.43
  - apple - house $\approx$ 36.87
- Angles between semantic vectors are inversely proportional to semantic similarity

Cosine distance

- Cosine of the angles between semantic vectors:
  - banana - apple = $\cos 26.56 \approx 0.89$
  - banana - house = $\cos 63.43 \approx 0.45$
  - apple - house = $\cos 36.87 \approx 0.80$
- Cosines of angles between semantic vectors are proportional to semantic similarity

Cosine distance

- Efficient computation of cosine distance:

$$\text{cosine}(\vec{w}_1, \vec{w}_2) = \frac{\vec{w}_1 \vec{w}_2}{||\vec{w}_1|| \; ||\vec{w}_2||} = \frac{\sum_{i=n}^{N} w_{1_i} w_{2_i}}{\sqrt{\sum_{i=n}^{N} {w_{1_i}}^2}\sqrt{\sum_{i=n}^{N} {w_{2_i}}^2}}$$

- Example:
  - $\overrightarrow{banana} : \langle 4, 2 \rangle$, $\overrightarrow{apple} : \langle 3, 4 \rangle$
  - $\text{cosine}(\overrightarrow{banana}, \overrightarrow{apple}) = \frac{\sum 4 * 3 + 2 * 4}{\sqrt{\sum 4^2 + 2^2}\sqrt{\sum 3^2 + 4^2}} \approx 0.89$

Reminder

	room	breakfast	school	painting	party
table	1	1	2	5	4
house	3	0	1	2	2
cat	1	0	0	0	1
banana	0	4	0	2	1
apple	0	3	1	4	0

Cosine similarity

	table	house	cat	banana	apple
table	1	...	...	...	...
house	0.79	1	...	...	...
cat	0.52	0.83	1	...	...
banana	0.57	0.31	0.15	1	...
apple	0.72	0.42	0	0.86	1

Cosine similarity: Laplace smoothing + PPMI

	table	house	cat	banana	apple
table	1	...	...	...	...
house	0.18	1	...	...	...
cat	0.39	0.94	1	...	...
banana	0	0	0	1	...
apple	0.31	0.02	0	0.79	1

Latent semantic analysis (LSA) (Landauer & Dumais, 1997)

- Co-occurrence matrix: term-document matrix
- Weighting scheme: division of local weight by global weight:
    - local weight: $\log(\mathrm{tf}_{w_1 w_2} + 1)$
    - global weight: $-\sum_{j=1}^{N} p(i,j) * \log p(i,j)$
      where N is the number of documents
- Dimension reduction: SVD (300 dimensions)
- Distance metric: cosine distance

Hyperspace analogue to language (HAL) (Lund & Burgess, 1996)

- Co-occurrence matrix: term-term matrix (window size: 10)
- Weighting scheme: none
- Dimension reduction: none or entropy based selection of columns
- Distance metric: Euclidean distance

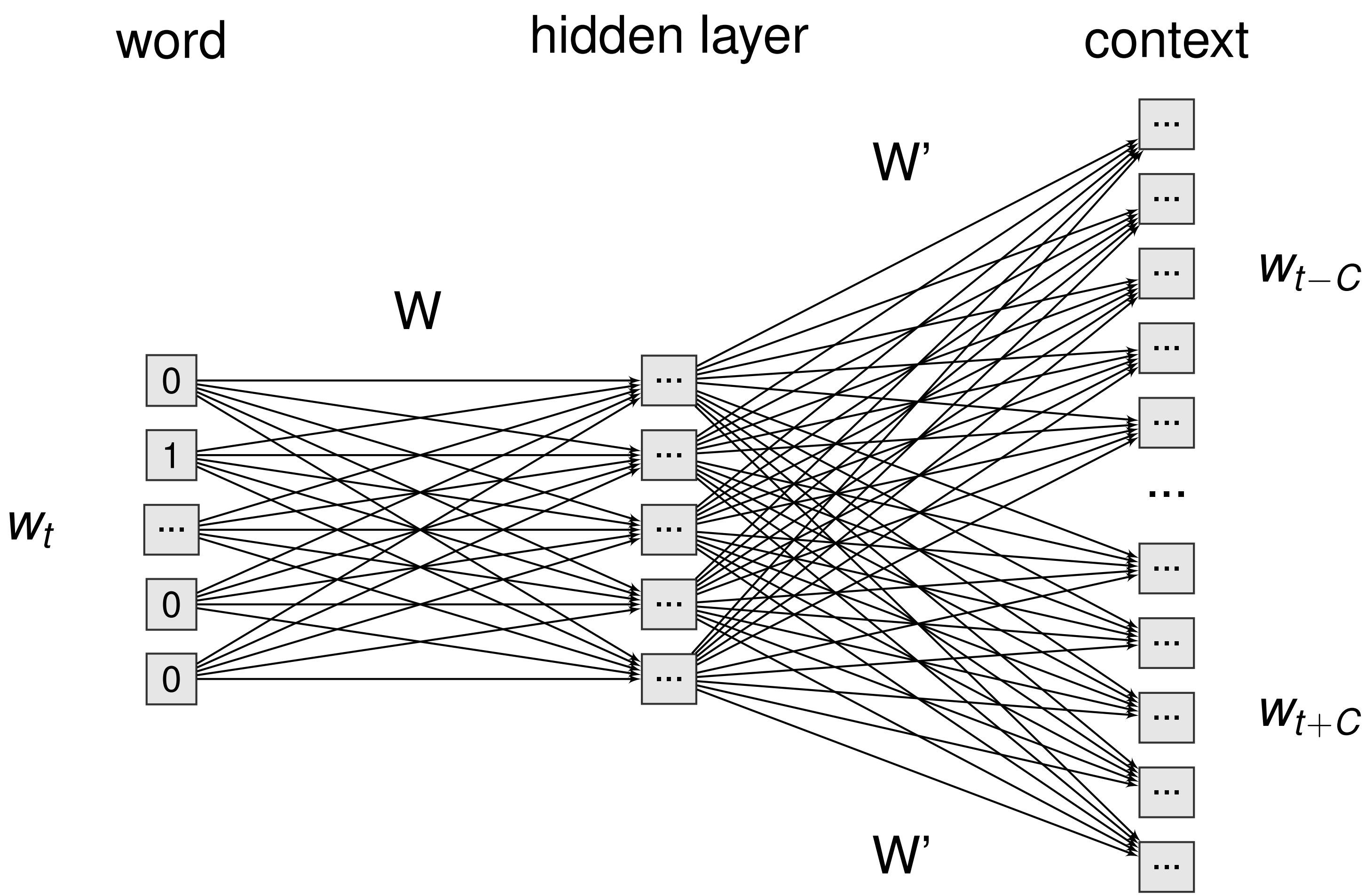
Predictive models

- Count-based models:
    - traditional type of model in vector semantics
    - semantic vectors derived from co-occurrence matrices
    - computationally expensive
- Predictive models:
    - current trend in vector semantics
    - semantic vectors based on predictions of target words or context words
    - computationally efficient

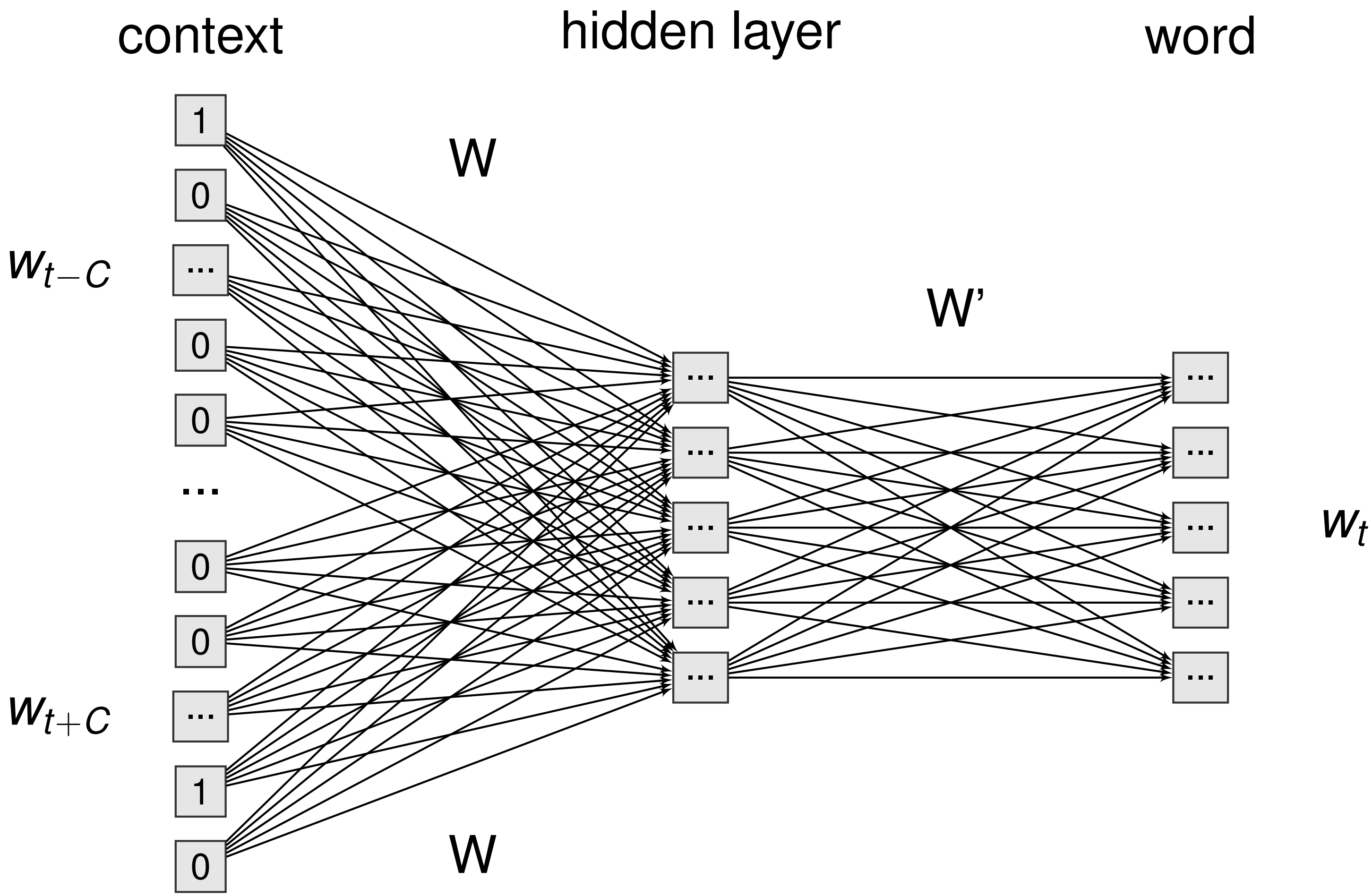
word2vec

- Predictive models: word2vec
  (Mikolov et al., 2013)
- Two types of models:
  - skip-gram (predict context words from target words)
  - CBOW (predict target words from context words)
- Both types of models are feed-forward neural networks

word2vec: skip-gram



word2vec: CBOW



Model objective

- Maximize the probabibility of the observed context words for all words in the corpus:

$$\underset{\theta}{\operatorname{argmax}} \prod_{w \epsilon T} \left[\prod_{c \epsilon C(w)} p(w_c | w_t; \theta) \right]$$

  where T is the corpus text and θ is the set of parameters associated with the weight matrices W and W'

  (Goldberg & Levy, 2014)

- How?

Algorithm

- For each word in the corpus:
  1) Present input (one-hot encoding)
  2) Calculate output activations given current W and W'
  3) Convert output activations to probabilities:

$$p(w_{c_i}|w_t) = \frac{\exp\left(act_i\right)}{\sum_{i=1}^{N} \exp\left(act_j\right)}$$

     where N is the number of word types in the corpus

  4) Compare output to actual output
  5) Adjust W and W' through back-propagation

Output

- Output: matrix W
- Semantic vectors are rows in W

Optimization

- Computational tricks:

    1) sub-sampling of frequent words
    2) optimization of probability estimation
    3) negative sampling

How to use word2vec?

```
# 1) Get word2vec
svn checkout http://word2vec.googlecode.com/svn/trunk/
# Or:
git clone https://github.com/dav/word2vec
#
# 2) Install word2vec
make
# 3) Run word2vec:
word2vec -train corpus.txt -output vectors.txt -cbow 0 -size 200
# Some parameters:
# -cbow (1 = cbow, 0 = skip-gram)
# -window (window size; skip length between words)
# -size (number of dimensions in semantic space)
# -threads (number of threads to use)
# ...
```

Reading semantic vectors into R

```
# Read file created by word2vec
# Created with skip-gram, window size of 5, 200 dimensions
vectors = read.table("vectors.txt",quote="",comment.char="",fill=TRUE)

# Remove first line (general information)
vectors = vectors[-1,]

# Define words
words = vectors[,1]

# Remove first column (words)
vectors = vectors[,-1]
```

Reading semantic vectors into R

```
# Turn into matrix
vectors = as.matrix(vectors)
rownames(vectors) = words
colnames(vectors) = 1:ncol(vectors)

# Look at dimensions
dim(vectors)
# [1] 552403    200

# Save matrix
save(vectors,file="vectors.rda")
```

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Cosine distance

```
# Load semantic vectors
load("data/vectors.rda")

# Define function to calculate cosine similarity
cos.dist <- function(word1,word2,matrix) {
  w1 = matrix[word1,]
  w2 = matrix[word2,]
  cos.dist = sum(w1*w2)/sqrt(sum(w1^2)*sum(w2^2))
  return(cos.dist)
}
```

Cosine distance

```
# Get semantic similarities
cos.dist(word1 = "apple",word2 = "banana", matrix = vectors)
# [1] 0.7725999
cos.dist(word1 = "apple",word2 = "house", matrix = vectors)
# [1] 0.04490028
cos.dist(word1 = "banana",word2 = "house", matrix = vectors)
# [1] 0.02936583
```

Semantic neighbors

```
# Load library for parallel processing
library(parallel)

# Define function to get semantic neigbhors
cos.dist.all <- function(word1,matrix,cores=4) {
  w1 = matrix[word1,]
  distances = unlist(mclapply(1:nrow(matrix),FUN = function(i) {
    w2 = matrix[i,]
    cos.dist = sum(w1*w2)/sqrt(sum(w1^2)*sum(w2^2))
    return(cos.dist)
  },mc.cores=cores))
  names(distances) = rownames(matrix)
  distances = rev(sort(distances))
  return(distances)
}
```

Semantic neighbors

```
# Get semantic neighbors
neighbors = cos.dist.all(word1 = "apple", matrix = vectors)
neighbors[1:10]
#       apple      almond     pumpkin   pineapple        pear
#   1.0000000   0.8486669   0.8482427   0.8440529   0.8420063
#     avocado  strawberry   blueberry       pecan pomegranate
#   0.8414042   0.8395716   0.8369698   0.8331580   0.8308719
```

Semantic neighbors

```
# Get semantic neighbors
neighbors = cos.dist.all(word1 = "banana", matrix = vectors)
neighbors[1:10]
#     banana     almond      mango  pineapple    coconut strawberry
#  1.0000000  0.7936759  0.7911360  0.7888999  0.7855046  0.7831195
#    avocado       pear      apple      melon
#  0.7775446  0.7727675  0.7725999  0.7714198
```

Semantic neighbors

```
# Get semantic neighbors
neighbors = cos.dist.all(word1 = "house", matrix = vectors)
neighbors[1:10]
#           house        bungalow five-bedroomed       townhouse
#       1.0000000       0.7794419       0.7021261       0.7019160
#       apartment   semi-detached    four-bedroom     mid-terrace
#       0.6956096       0.6914265       0.6902554       0.6888919
#     five-bedroom   three-bedroom
#       0.6817849       0.6805665
```

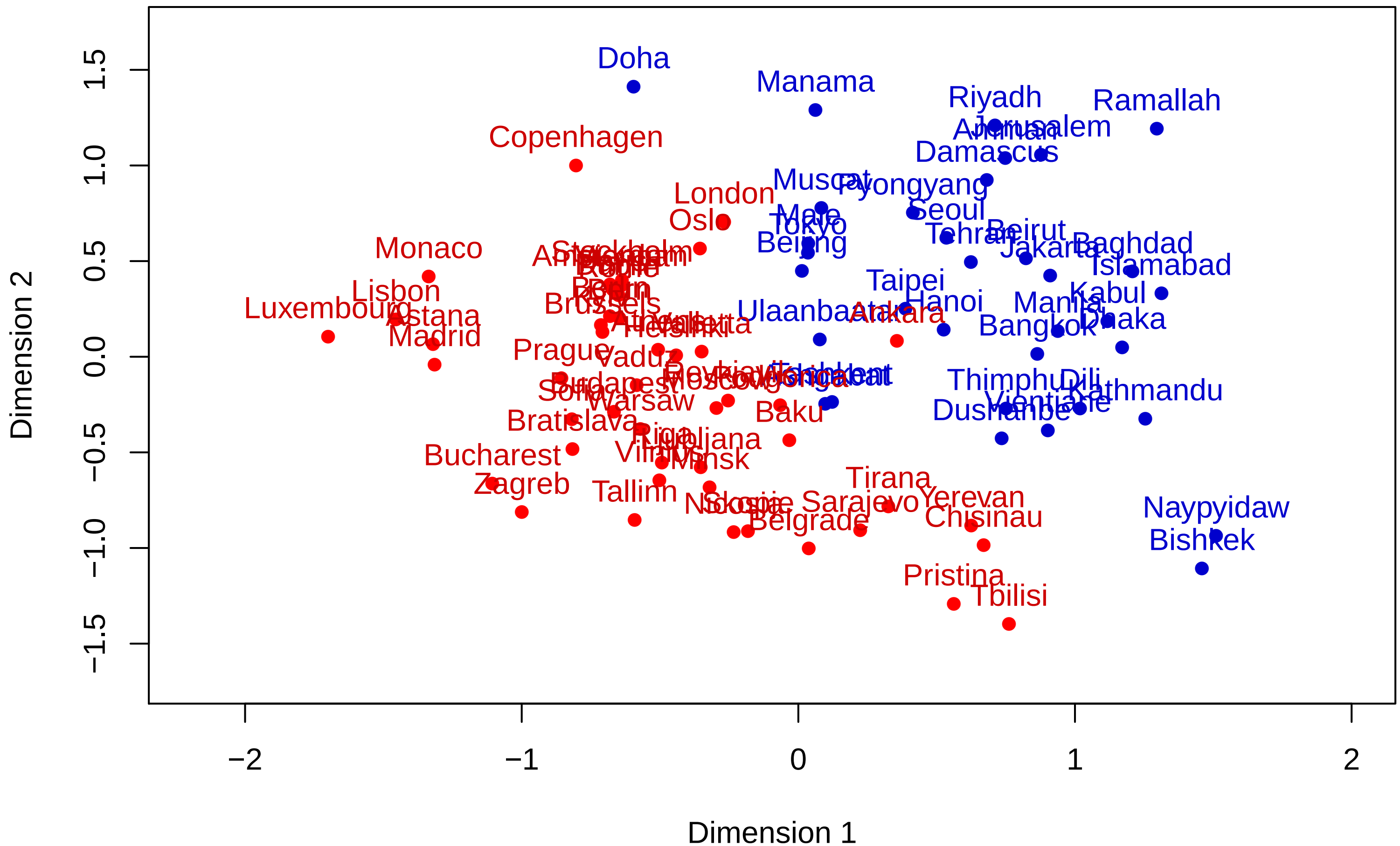
Semantic clusters

- Automatically learn semantic categorization:
  1) extract relevant semantic vectors
  2) apply clustering technique

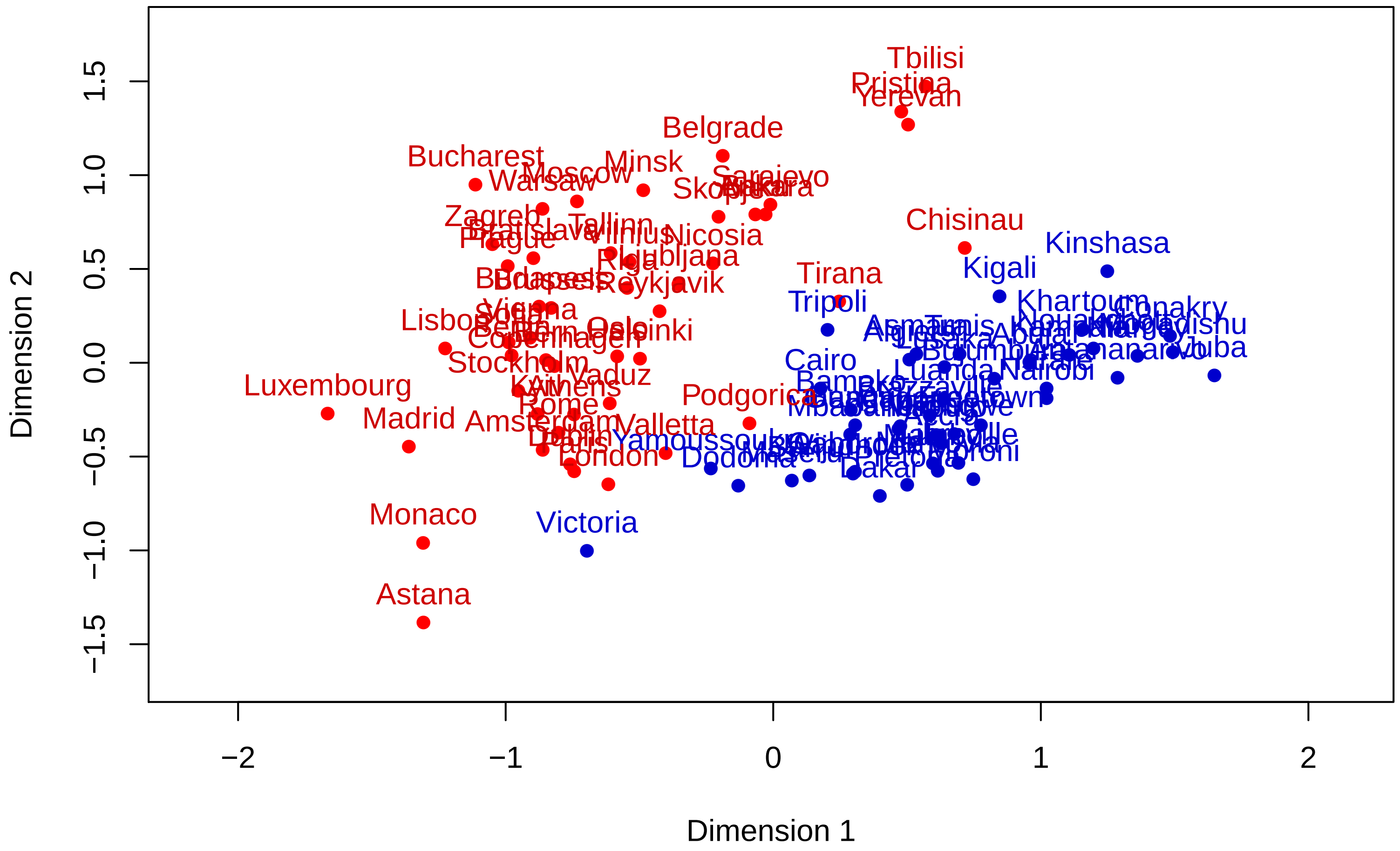
Semantic clusters

- Example:
  1) capitals of continents Europe, Asia and Africa
  2) multidimensional scaling:
     - calculate Euclidean distance between all points
     - represent distance information in k dimensions (here: $k = 2$)

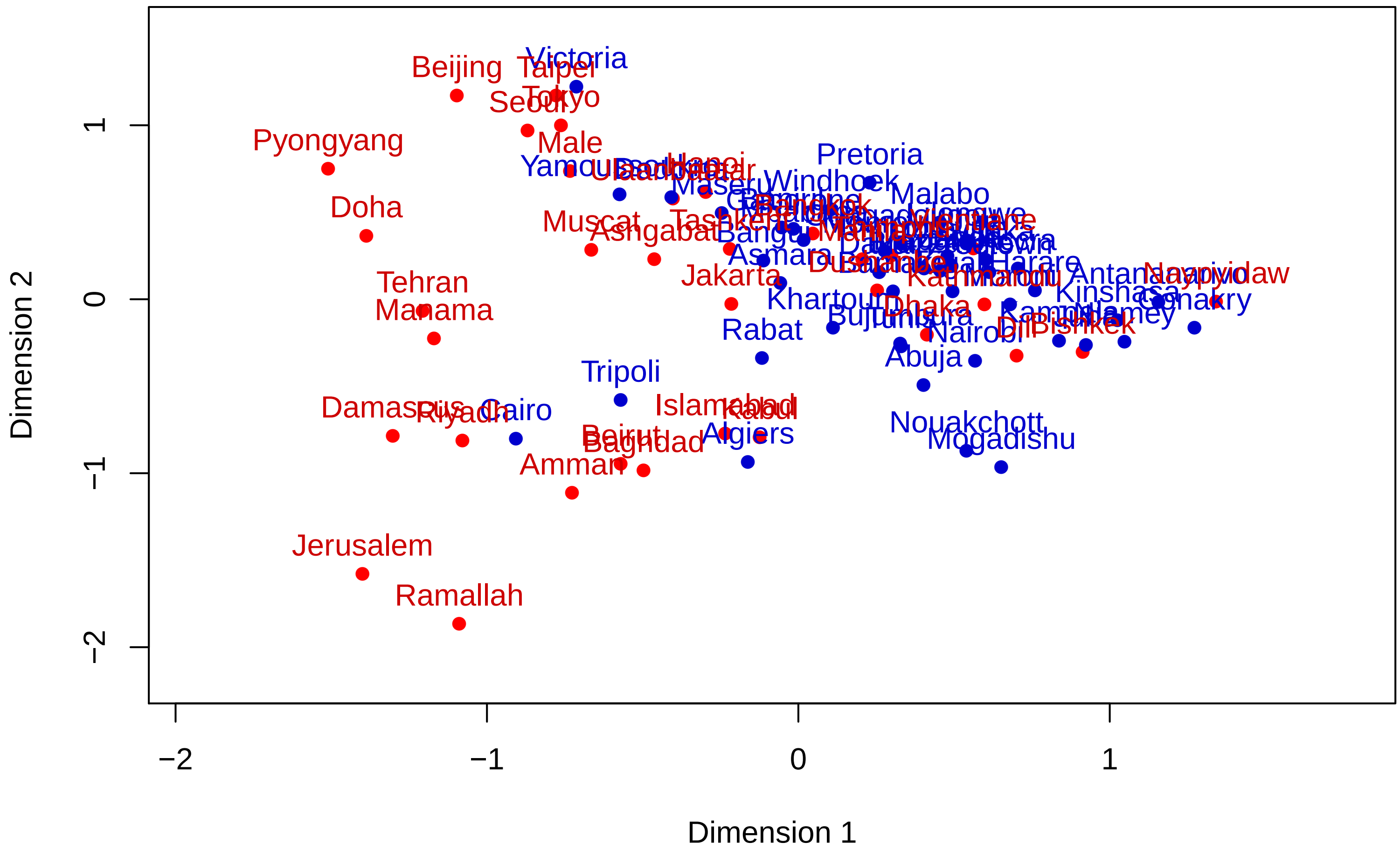
Semantic clusters



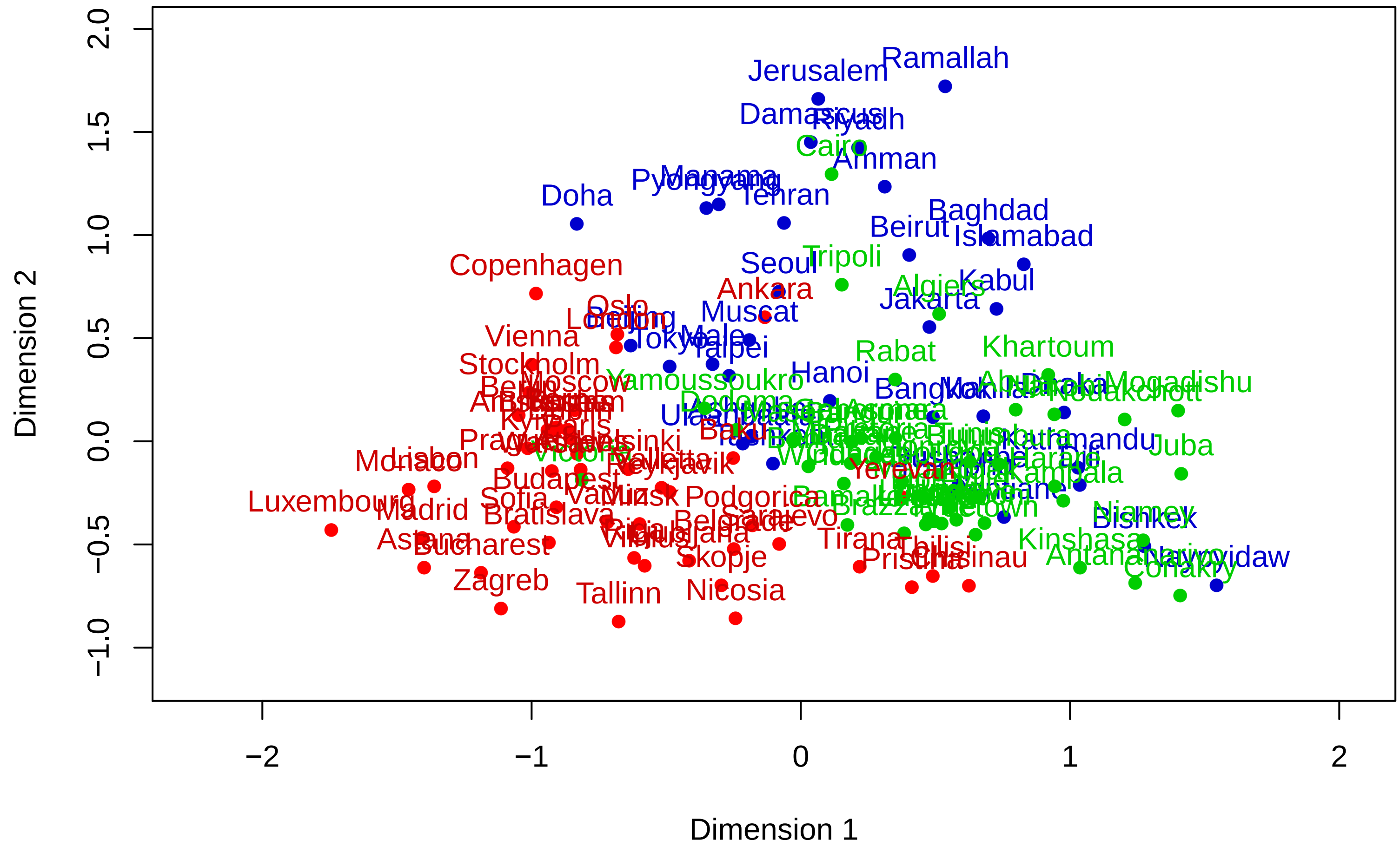
Semantic clusters



Semantic clusters



Semantic clusters



Evaluation

- Evaluation of semantic space models
  - correlation with human ratings
  - semantic similarity tasks (e.g.; synonym detection)
  - analogy test

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Analogy test

- Semantic and syntactic analogies
  - Semantic: Greece : Athens :: China : . . . ?
  - Syntactic: good : better :: hard : . . . ?
- Model answer:
  1) calculate $\vec{v} = \overrightarrow{Athens} - \overrightarrow{Greece} + \overrightarrow{China}$
  2) return closest semantic neighbor of $\vec{v}$

Analogies: function

```r
# Define function to get semantic neighbors
cos.dist.analogy <- function(words,matrix,cores=4,nrows=30000) {
  w1 = matrix[words[2],] - matrix[words[1],] + matrix[words[3],]
  matrix = matrix[1:nrows,]
  distances = unlist(mclapply(1:nrow(matrix),FUN = function(i) {
    w2 = matrix[i,]
    cos.dist = sum(w1*w2)/sqrt(sum(w1^2)*sum(w2^2))
    return(cos.dist)
  },mc.cores=cores))
  names(distances) = rownames(matrix)
  distances = rev(sort(distances))
  distances = distances[which(!(names(distances)%in%words))]
  return(distances)
}
```

Analogies: semantic

```
# Get semantic neighbors
answer = cos.dist.analogy(words = c("Greece","Athens","China"),
                          matrix = vectors)[1]
answer
#  Beijing
# 0.774342
```


Analogies: semantic

```
# Get semantic neighbors
answer = cos.dist.analogy(words = c("Italy","Rome","Germany"),
                          matrix = vectors)[1]
answer
#    Berlin
# 0.6691689
answer = cos.dist.analogy(words = c("Norway","Oslo",
                          "Germany"),matrix = vectors)[1:3]
answer
# Frankfurt    Prague    Berlin
# 0.5891469 0.5633364 0.5556203
```

Analogies: syntactic

```
# Get semantic neighbors
answer = cos.dist.analogy(words = c("good","better","hard"),
                          matrix = vectors)[1]
answer
#    harder
# 0.7518293
answer = cos.dist.analogy(words = c("good","better","great"),
                          matrix = vectors)[1:3]
answer
#    bigger   greater   quicker
# 0.6109970 0.5557460 0.5388236
```

Types of analogies

category	example
capital	Accra : Ghana :: Antananarivo : . . .
state	Chicago : Illinois :: Houston : . . .
adverb	amazing : amazingly :: quick : . . .
opposite	certain : uncertain :: rational : . . .
comparative	good : better :: hard : . . .
superlative	big : biggest :: good : . . .
participle	dance : dancing :: go : . . .
country adjective	England : English :: Korea : . . .
past tense	looking : looked :: seeing : . . .
plural	hand : hands :: child : . . .
3^{rd} person singular	predict : predicts :: see : . . .

Performance

- Compare performance of semantic vector models when trained on a 6 billion word corpus: (Pennington, Socher & Manning, 2014)

model	performance
SVD, no weighting	7.3%
SVD, $\log(1 + f_{ij})$ weighting	60.1%
word2vec: CBOW	65.7%
word2vec: skip-gram	69.1%

Performance

- Are predictive models better than count-based models?
- Not necessarily:
  - strong mathematical relationship between word2vec and PPMI weighting
  - implementation and hyperparameters may play a large role

Global vectors (GloVe) (Pennington, Socher & Manning, 2014)

- New count-based model
- Idea: the ratio of co-occurrence probabilities provides semantic information

probability	solid	gas	water	fashion
$P(k\|ice)$	large	small	large	small
$P(k\|steam)$	small	large	large	small
$P(k\|ice)/P(k\|steam)$	>1	<1	$\sim$1	$\sim$1

- Frequency weighting: $\log(f_{w_1 w_2} + 1)$
- Adapt loss function to calculate semantic vectors that satisfy ratios of co-occurrence probabilities

Performance GloVe

- Compare performance of semantic vector models when trained on a 6 billion word corpus:
(Pennington, Socher & Manning, 2014)

model	performance
SVD, no weighting	7.3%
SVD, $\log(1 + f_{ij})$ weighting	60.1%
word2vec: CBOW	65.7%
word2vec: skip-gram	69.1%
GloVe	71.7%

Conclusions

- Vector semantics provides numerical estimates of meaning
- Solution to “poverty of the stimulus” problem
- Implications of individual differences?
- Ongoing battle between count-based and predictive models
- Applications:
    - cognitive models of semantics
    - computer science applications

Conclusions

Thank you