# Natural Language Processing IN2361

## Prof. Dr. Georg Groh

# Chapter 6
# Vector Semantics and Embeddings

- content is based on [1]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form  from [1]
- citations of [1] or from [1]  are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

# Lexical Semantics

- "meaning" of a word == ? aspects of "meaning" desirable to model:
  - general conceptual relatedness / similarity (dog, cat) (buy, sell, pay)
  - specific forms of conceptual relations: antonmy (hot, cold), hyponymy (cat, mammal)
  - distinguish between word senses ($bank^1$, $bank^2$) ($mouse^1$, $mouse^2$) of polysemous lemmas
  - …

# Word Senses

- Lemma is polysemous: it has multiple semantically unrelated word senses e.g.

  - lemma jaguar: jaguar[1] (the cat), jaguar[2] (the car) jaguar[3] (the Fender guitar model)

  - lemma bank: bank[1] ("financial institution"), bank[2] ("sloping mound")

  - lemma bat:  bat[1] ("club for hitting a ball"), bat[2] ("nocturnal flying animal"):

  - each of those: homonyms and homographs (same writing)

- *write - right*;  *piece - peace:* homophones. ($\leftrightarrow$ spelling errors)

- homographs that are not homophones:
  bass[1] ("fish") - bass[2] ("instrument")
  ($\leftrightarrow$ speech synthesis errors)

- (nearly) identical senses of different lemmas: synonymy:
  - substitutable one for the other in any sentence without changing the truth conditions of the sentence (same propositional meaning)
  - *couch/sofa  vomit/throw up  filbert/hazelnut  car/automobile*

- synonymy: actually between senses of words: synonyms may replace one another in a sentence:
  - example:  big / large replaceable (big = big$^1$):
    - *How big is that plane?*
    - *Would I be flying on a large or small plane?*
  - big / large not replaceable (big = big$^2$):
    - *Miss Nelson became a kind of big sister to Benjamin.*

- synonyms: principle of contrast: different word forms → at least SLIGHTLY different meaning / different pragmatics.
  examples: water, $H_2O$ ;   car, automobile

- **Hyponmy**: sub-class - class relation;
  **Hypernymy** (Superordinate): super-class – class relation;
  - sense A  hyponym of  sense B  ("B **subsumes** A" "A **is-a** B")  if
    $\forall x: A(x) \rightarrow B(x)$
  - examples: 

| Hypernym: | vehicle | fruit | furniture | mammal |
|-----------|---------|-------|-----------|--------|
| Hyponym:  | car     | mango | chair     | dog    |

---

- **Meronym**: part-of - whole relation;
  **Holonym**: whole – part-of relation;
  - examples:

| Meronym:  | leg   | leg   | wheel | CPU      |
|-----------|-------|-------|-------|----------|
| Holonym:  | chair | human | car   | computer |

---

- **Antonymy**:
  - *long/short  big/little  fast/slow  cold/hot*  (opposite ends of a scale)
  - *rise/fall  up/down  in/out*  (reversed direction (reversives))

# Word Similarity, Word Relatedness

- general similarity between words (abstracting from word senses)

  - e.g. measure via human judgement (e.g. via crowd sourcing)

| | | |
|---|---|---|
| vanish | disappear | 9.8 |
| behave | obey | 7.3 |
| belief | impression | 5.95 |
| muscle | bone | 3.65 |
| modest | flexible | 0.98 |
| hole | agreement | 0.3 |

- word relatedness (association): words belong to the same semantic field, e.g. semantic field of

  - hospitals (surgeon , scalpel , nurse , anaesthetic , hospital ),
  - restaurants (waiter , menu , plate , food , chef ),
  - houses (door , roof , kitchen , family , bed ).

  semantic fields ←→ topics in Topic Models (Latent Dirchlet Allocation (LDA))

# Semantic Frames

- Semantic Frame:  special form of semantic field:
  set of words that denote perspectives or participants in a particular
  type of event.

- in semantic frame exist distinct Semantic Roles:

  - e.g. in semantic frame 'transaction of goods or services':
    buy or sell → buyer, seller; pay ←→ buyer etc.

- identifying association between words and roles → better sentence
  understanding (e.g. for question answering):
  Sam bought the book from Ling
  → == Ling sold the book to Sam,

# Connotation

- words can have affective meaning (connotation):
  sentiment, opinions, evaluations
  e.g. *happy* ←→ *sad*;  *great, love* ←→ *terrible, hate*

- dimensions of affective meaning:

  - valence: the pleasantness of the stimulus,
    e.g. *happy* ←→ *sad*

  - arousal: the intensity of emotion provoked by the stimulus,
    e.g. *excited* ←→ *relaxed*

  - dominance: the degree of control exerted by the stimulus
    e.g. *important, controlling* ←→ *awed, influenced*

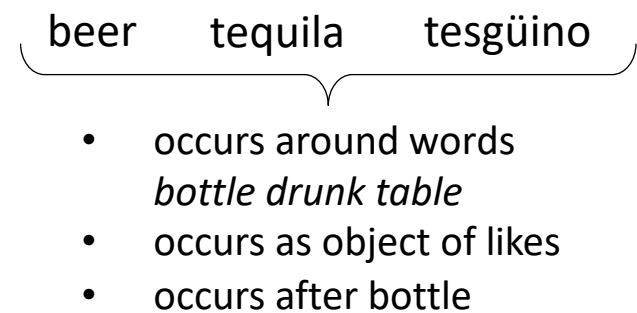|  | Valence | Arousal | Dominance |
|---|---|---|---|
| courageous | 8.05 | 5.5 | 7.38 |
| music | 7.67 | 5.57 | 6.5 |
| heartbreak | 2.45 | 5.65 | 3.58 |
| cub | 6.71 | 3.95 | 4.24 |
| life | 6.68 | 5.59 | 5.89 |

→ word semantics
as vectors

# Distributional Semantics

- "words that occur in similar contexts tend to have similar meanings"

- "you shall know a word by the company it keeps"

- *A bottle of tesgüino is on the table.*
  *Everybody likes tesgüino.*
  *Tesgüino  makes you drunk.*
  *We make tesgüino  out of corn.*

  beer      tequila      tesgüino

  - occurs around words
    *bottle drunk table*
  - occurs as object of likes
  - occurs after bottle

Ongchoi is delicious sauteed with garlic.
Ongchoi is superb over rice.
...ongchoi leaves with salty sauces...

...spinach sauteed with garlic over rice...
...chard stems and leaves are delicious...
...collard greens and other salty leafy greens

- distributional semantics:  meaning of a word is computed from the distribution of words around it

# Embeddings

- combine distributionalist intuition with vector representation idea → word embeddings
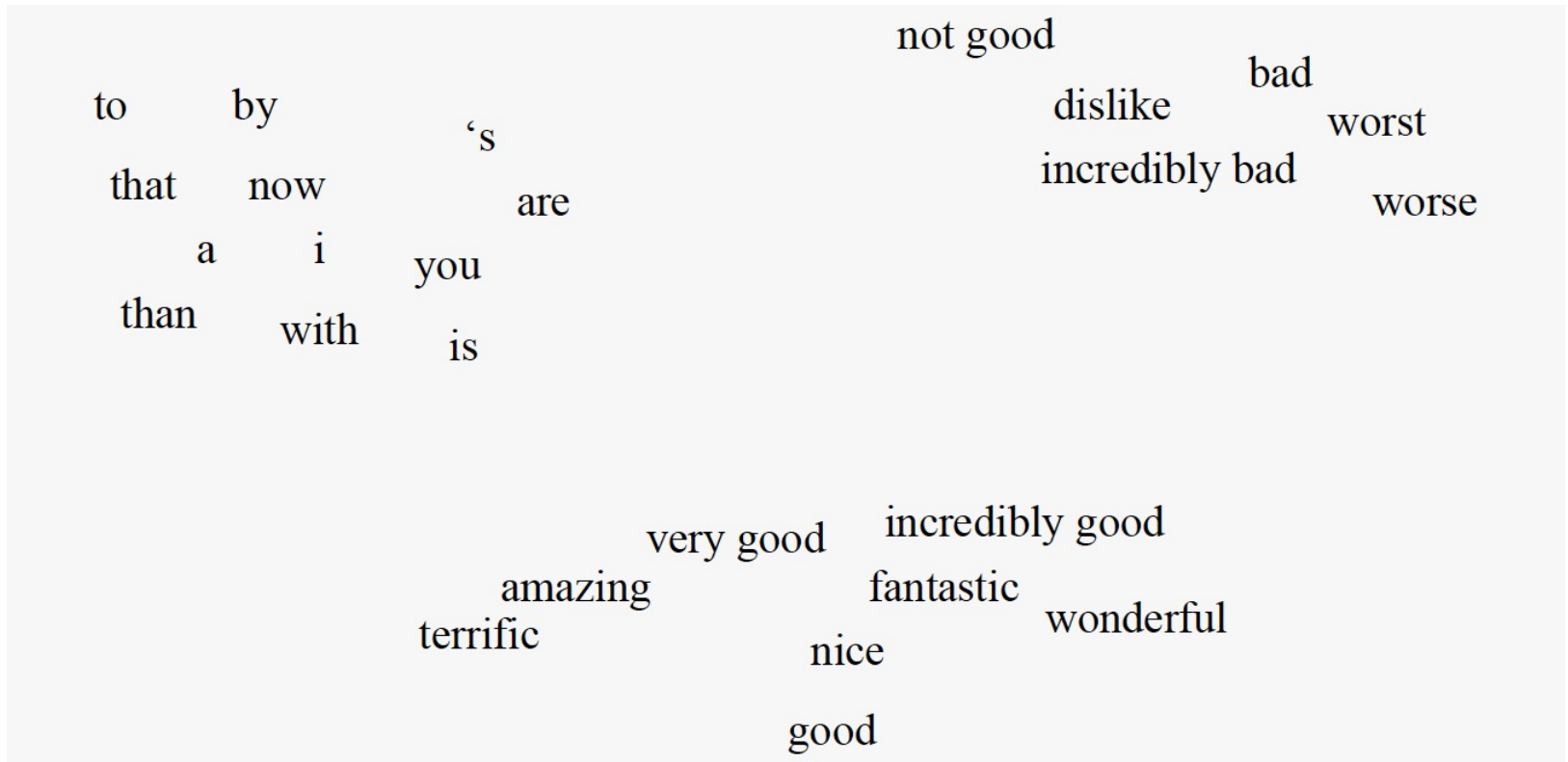
# Words and Vectors

- vector space models of semantics: model word by embedding it into a vector space → vector representations of a word: embedding

- semantic similarity of words ←→ similarity of vectors

- $|V|\times D$ term (word)-document matrix with $D$ document vectors:

|  | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| **battle** | 1 | 0 | 7 | 13 |
| **good** | 114 | 80 | 62 | 89 |
| **fool** | 36 | 58 | 1 | 4 |
| **wit** | 20 | 15 | 2 | 3 |
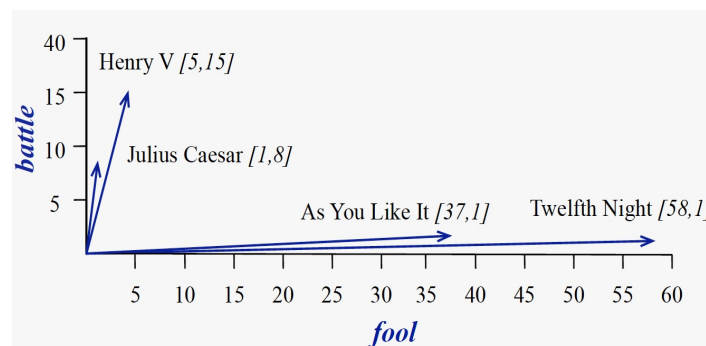


- Information retrieval: query $q$, document collection $D = \{d_1, d_2, \dots\}$: find $\{d_i\}$ with high $sim(d_i, q)$. → simple application of tf-idf document (column) vectors : $sim(d_i, q) = \cos(d_i, q) = (||d_i|| \, ||q||)^{-1} \mathbf{d}_i^T \boldsymbol{q}$

# Words as Vectors

- **words as vectors**: e.g. **rows** of **word-document** matrix:
  *fool*:[37,58,1,5],  *clown*:[5, 117, 0, 0],  *battle*:[1,1,8,15]

- **more common:** word-vectors: rows of $|V| \times |V|$ **word-word** matrix (term-term matrix, term-context matrix)

  - **matrix elements**: feature-values (e.g. counts) of co-occurences in certain **contexts**: documents, n-word window around word,...

| | aardvark | ... | computer | data | result | pie | sugar | ... |
|---|---|---|---|---|---|---|---|---|
| **cherry** | 0 | ... | 2 | 8 | 9 | 442 | 25 | |
| **strawberry** | 0 | ... | 0 | 0 | 1 | 60 | 19 | |
| **digital** | 0 | ... | 1670 | 1683 | 85 | 5 | 4 | |
| **information** | 0 | ... | 3325 | 3982 | 378 | 5 | 13 | |

# Words as Vectors
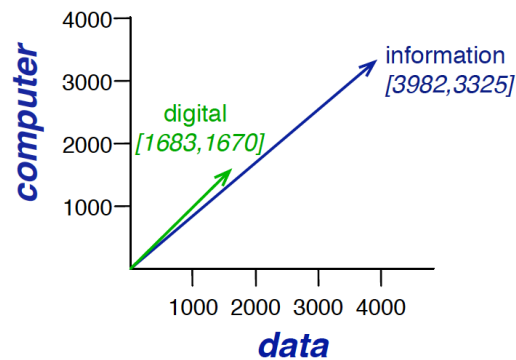
- question: instead of counts (which may not be very discriminative / informative): what features can we use? → TF-IDF, Pointwise Mutual Information etc.

- other question: how to compute similarity in a vector space? → usually use cosine similarity

$$\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$
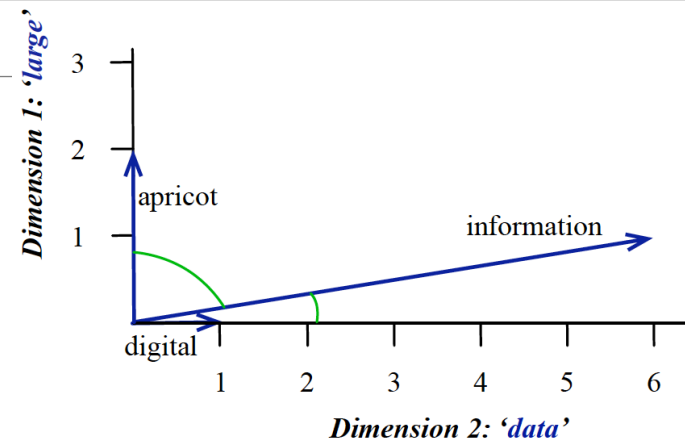
# Similarity Measures on Word-Vectors

$$\text{sim}_{\text{Cosine}}(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}||\vec{w}|}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} \max(v_i, w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) = \frac{2 \times \sum_{i=1}^{N} \min(v_i, w_i)}{\sum_{i=1}^{N} (v_i + w_i)}$$

$$\text{dist}_{\text{Jensen-Shannon}}(\vec{v}, \vec{w}) = D(\vec{v}\|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}\|\frac{\vec{v}+\vec{w}}{2})$$

JS∈[0,1] → e.g. $\text{sim}_{\text{JS}} = 1 - \text{dist}_{\text{JS}}$

*Dimension 1: 'large'*

apricot

information

digital

1   2   3   4   5   6

*Dimension 2: 'data'*

KL-divergence: $D(P\|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$

$$JS(P\|Q) = D(P\|\frac{P+Q}{2}) + D(Q\|\frac{P+Q}{2})$$

# TF-IDF

- **term frequencies** alone: **not very discriminative**
  (e.g. regard frequent words such as the, I, they, good

- **→ alternative term-document** matrix:
  **tf-idf**: term frequency in a document $\mathrm{tf}_{t,d} = \mathrm{count}(t,d)$

  * inverse document frequency $\mathrm{idf}_t = \log_{10}\left(\dfrac{N}{\mathrm{df}_t}\right)$

  $N$ : number of documents

  $\mathrm{df}_t$ : number of documents in which t occurs

**→** matrix element for word $t$ and document $d$:

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t$$

| Word | df | idf |
|------|-----|-------|
| Romeo | 1 | 1.57 |
| salad | 2 | 1.27 |
| Falstaff | 4 | 0.967 |
| forest | 12 | 0.489 |
| battle | 21 | 0.074 |
| fool | 36 | 0.012 |
| good | 37 | 0 |
| sweet | 37 | 0 |

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|--------|------|------|--------|--------|
| **battle** | 0.074 | 0 | 0.22 | 0.28 |
| **good** | 0 | 0 | 0 | 0 |
| **fool** | 0.019 | 0.021 | 0.0036 | 0.0083 |
| **wit** | 0.049 | 0.044 | 0.018 | 0.022 |

# Applications of TF-IDF

- applications of tf-idf + cosine similarity:
  e.g. information retrieval, plagiarism detection, recommender systems:

- idea: represent document by centroids of its word vectors:

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

then compute $sim(d_1, d_2) = \cos(d_1, d_2)$

# Pointwise Mutual Information

- **Specific Conditional Entropy:**

$$H(X|Y=y) = -\sum_x P(X=x|Y=y)\log_2 P(X=x|Y=y)$$

- **Conditional Entropy:**

$$H(X|Y) = -\sum_y P(Y=y)\ H(X|Y=y)$$

- **Mutual Information:**

$$I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

$$= \sum_x \sum_y P(X=x, Y=y)\log_2 \frac{P(X=x, Y=y)}{P(X=x)P(Y=y)}$$

- **Pointwise Mutual Information (PMI):**

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

how often two events x and y occur, compared with what we would expect if they were independent

# Pointwise Mutual Information

- PMI between a word w and a word c in context:

$$I(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}$$

co-occurrence of words (MLE)

expectation of co-occurrence if words were independent

- $I(w,c) \in [-\infty, \infty]$ but negative values (co-occurrence *less* likely than independent occurrences) not very reliable →
Positive PMI:

$$\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

- Assuming a co-occurrence count matrix F, we have

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

# Pointwise Mutual Information

|  | computer | data | result | pie | sugar | count(w) |
|---|---|---|---|---|---|---|
| cherry | 2 | 8 | 9 | 442 | 25 | 486 |
| strawberry | 0 | 0 | 1 | 60 | 19 | 80 |
| digital | 1670 | 1683 | 85 | 5 | 4 | 3447 |
| information | 3325 | 3982 | 378 | 5 | 13 | 7703 |
| count(context) | 4997 | 5673 | 473 | 512 | 61 | 11716 |

| | p(w,context) | | | | | p(w) |
|---|---|---|---|---|---|---|
| | computer | data | result | pie | sugar | p(w) |
| cherry | 0.0002 | 0.0007 | 0.0008 | 0.0377 | 0.0021 | 0.0415 |
| strawberry | 0.0000 | 0.0000 | 0.0001 | 0.0051 | 0.0016 | 0.0068 |
| digital | 0.1425 | 0.1436 | 0.0073 | 0.0004 | 0.0003 | 0.2942 |
| information | 0.2838 | 0.3399 | 0.0323 | 0.0004 | 0.0011 | 0.6575 |
| p(context) | 0.4265 | 0.4842 | 0.0404 | 0.0437 | 0.0052 | |

$$P(\text{w=information,c=data}) = \frac{3982}{11716} = .3399$$

$$P(\text{w=information}) = \frac{7703}{11716} = .6575$$

$$P(\text{c=data}) = \frac{5673}{11716} = .4842$$

$$\text{ppmi(information,data)} = \log 2(.3399/(.6575*.4842)) = .0944$$

| PPMI | computer | data | result | pie | sugar |
|---|---|---|---|---|---|
| cherry | 0 | 0 | 0 | 4.38 | 3.30 |
| strawberry | 0 | 0 | 0 | 4.10 | 5.51 |
| digital | 0.18 | 0.01 | 0 | 0 | 0 |
| information | 0.02 | 0.09 | 0.28 | 0 | 0 |

- **PMI biased towards infrequent events** →

    ○ use $\text{PPMI}_\alpha$:

$$\text{PPMI}_\alpha(w,c) = \max\left(\log_2 \frac{P(w,c)}{P(w)P_\alpha(c)}, 0\right) \qquad P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

e.g. with $\alpha = 0.75$ → $P_\alpha(c) > P(c)$ → PPMI smaller

    ○ use **Laplace Smoothing**:

| | computer | data | pinch | result | sugar |
|---|---|---|---|---|---|
| apricot | 0 | 0 | 1 | 0 | 1 |
| pineapple | 0 | 0 | 1 | 0 | 1 |
| digital | 2 | 1 | 0 | 1 | 0 |
| information | 1 | 6 | 0 | 4 | 0 |

| PPMI | computer | data | pinch | result | sugar |
|---|---|---|---|---|---|
| apricot | 0 | 0 | 2.25 | 0 | 2.25 |
| pineapple | 0 | 0 | 2.25 | 0 | 2.25 |
| digital | 1.66 | 0 | 0 | 0 | 0 |
| information | 0 | 0.57 | 0 | 0.47 | 0 |

| | computer | data | pinch | result | sugar |
|---|---|---|---|---|---|
| apricot | 2 | 2 | 3 | 2 | 3 |
| pineapple | 2 | 2 | 3 | 2 | 3 |
| digital | 4 | 3 | 2 | 3 | 2 |
| information | 3 | 8 | 2 | 6 | 2 |

| PPMI | computer | data | pinch | result | sugar |
|---|---|---|---|---|---|
| apricot | 0 | 0 | 0.56 | 0 | 0.56 |
| pineapple | 0 | 0 | 0.56 | 0 | 0.56 |
| digital | 0.62 | 0 | 0 | 0 | 0 |
| information | 0 | 0.58 | 0 | 0.37 | 0 |

# Dense Word-Vectors

- in a $|V| \times |V|$ word-word-matrix, each $|V|$-dim. word vector (with $|V| \propto 10^5$) will be sparse → use more dense, lower dim. vectors with dim $\propto 10^3$:

    ○ models will have to learn fewer parameters, less overfitting, better modelling of semantic relations

    ○ if dimensions (say for car and automobile) are distinct: sparse, high-dim. vectors may fail to model the semantic similarity between words often occurring together with car and automobile

- first idea: use dimensionality reduction techniques (SVD, PCA, pPCA, Factor-Analysis, Auto-Encoders etc.) → see [5] and / or slides in appendix in this slide-set

- modern idea: use prediction based embeddings (word2vec, GloVe etc.)

- **intuition** of word2vec: instead of counting how often each word w occurs near e.g. *apricot* , train a classifier on a binary prediction task: "Is word w  likely to show up near apricot ?"

- use classifier weights as dense embeddings for words

- use large corpora as "supervised" data sources for classifier

1. Treat the target word and a neighboring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the regression weights as the embeddings

# (Log. Regression) Classifier

```
... lemon,   a [tablespoon of apricot jam,      a] pinch ...
                   c1              c2   w   c3          c4
```

$$Similarity(w,c) \approx w \cdot c$$

word
w

any word c
in context
window

vector
repre-
sentation
of w

vector
repre-
sentation
of c

$$P(+|w,c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c} \cdot \mathbf{w})}$$

$$P(-|w,c) = 1 - P(+|w,c)$$

$$= \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1+\exp(\mathbf{c} \cdot \mathbf{w})}$$

- assuming independence between context words, we get

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(\mathbf{c_i} \cdot \mathbf{w})$$

# Learning Skip-Gram Embeddings

```
... lemon,   a [tablespoon of apricot jam,       a] pinch ...
                 c1              c2      w     c3        c4
```

**positive examples +**

| $w$ | $c_{pos}$ |
|---|---|
| apricot | tablespoon |
| apricot | of |
| apricot | jam |
| apricot | a |

**negative examples -**

| $w$ | $c_{neg}$ | $w$ | $c_{neg}$ |
|---|---|---|---|
| apricot | aardvark | apricot | seven |
| apricot | my | apricot | forever |
| apricot | where | apricot | dear |
| apricot | coaxial | apricot | if |

for each positive example,
k negative examples (here k=2)

- sample negative examples from modified unigram distribution:

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_{w'} count(w')^\alpha}$$

usually with $\alpha = 0.75$ (give rare words higher probability)

# Learning Skip-Gram Embeddings

- → objective function:

$$L(\theta) = -\left[ \sum_{(w,c)\,\in\,+} \log P(+|w,c) + \sum_{(w,c)\,\in\,-} \log P(-|w,c) \right]$$

for **one** word - context pair (w,c) and k noise words:

$$L_{CE}(\theta) = -\log\left[ P(+|w,c_{pos}) \prod_{i=1}^{k} P(-|w,c_{neg_i}) \right]$$

$$= -\left[ \log P(+|w,c_{pos}) + \sum_{i=1}^{k} \log P(-|w,c_{neg_i}) \right]$$

$$\theta = \{w, c_{pos}, \{c_{neg_i}\}\}$$

←→ maximize vector similarity of (w,c) drawn from positive examples, minimize for negative examples

- start with random embeddings;
  use stochastic gradient descent as usual for log. regression;
  k, dimension d, and window size L: meta parameters

- → we get two different embeddings for each word:
  - target embedding w (matrix W)
  - context embedding c (matrix C)



decrease
similarity( apricot , aardvark)
$$w_i \cdot c_k$$

increase
similarity( apricot , jam)
$$w_i \cdot c_j$$

W

1... ... d

apricot
target word
1
i
.
.
.
.
.
.
V

"…apricot jam…"

1.2.......j.................k......V

1
.
.
.
.
d

C

jam
neighbor word

aardvark
random noise word
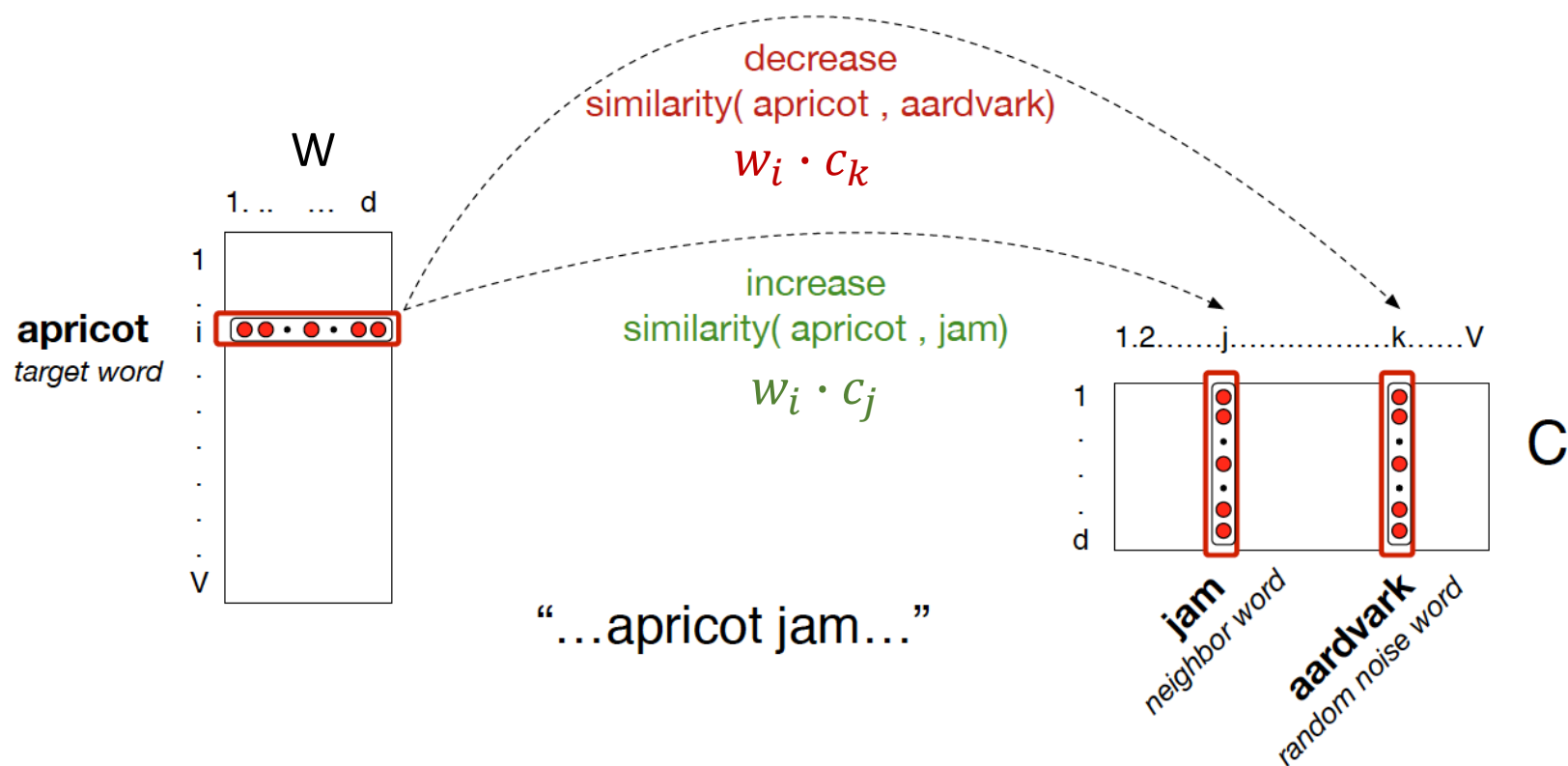
- final embeddings: use only w, or add w and c, or stack them

# Visualizing Embeddings

- use word clouds of most similar words

- or use projection technique (PCA or t-SNE)
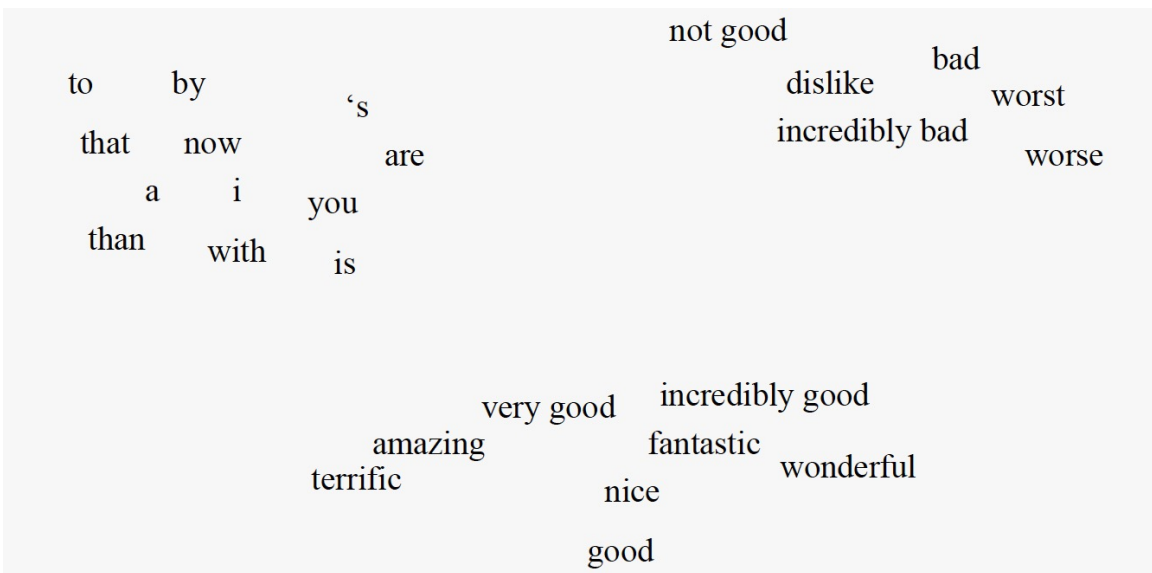


- or use hierarchical clustering
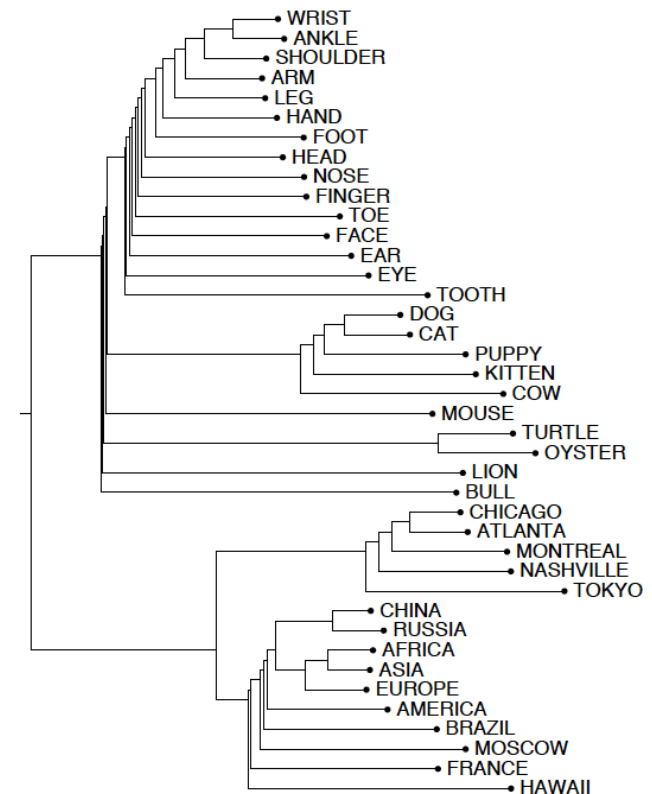
- **short context windows**: most similar words: **semantically similar** words; **long context windows**: most similar words: **topically related** words

  - *Hogwarts* and $L = \pm 2$ → other „schools": *Sunnydale* (from Buffy the Vampire Slayer) or *Evernight* (from a vampire series)

  - *Hogwarts* and $L = \pm 5$ → *Dumbledore*, *Malfoy*, and *half-blood*

- **first order** co-occurence (**syntagmatic** co-occurrence): words that typically occur near each other: *wrote* ←→ *book, poem* **second order** co-occurrence (**paradigmatic** co-occurrence): words that have similar neighbours: *wrote* ←→ *said, remarked*

- skip-gram embeddings show semantic and other forms of compositionality:
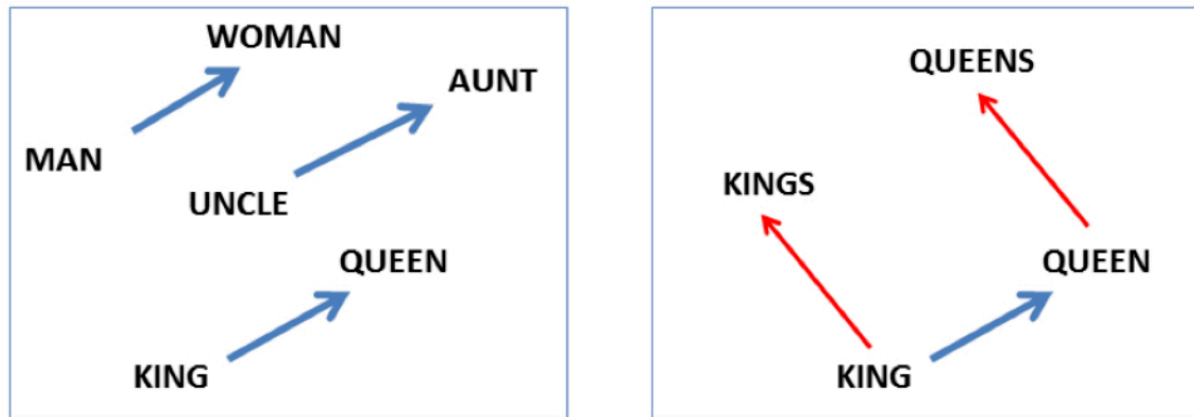


2D PCA projection of 1000-dim. embeddings:
"king" – "man" + "woman" ≈ "queen"
"paris" – "France" + "Italy" ≈ "Rome"

| Czech + currency | Vietnam + capital | German + airlines | Russian + river | French + actress |
|---|---|---|---|---|
| koruna | Hanoi | airline Lufthansa | Moscow | Juliette Binoche |
| Check crown | Ho Chi Minh City | carrier Lufthansa | Volga River | Vanessa Paradis |
| Polish zolty | Viet Nam | flag carrier Lufthansa | upriver | Charlotte Gainsbourg |
| CTK | Vietnamese | Lufthansa | Russia | Cecile De |

closest 4 tokens of sum of skip-gram embeddings [6]
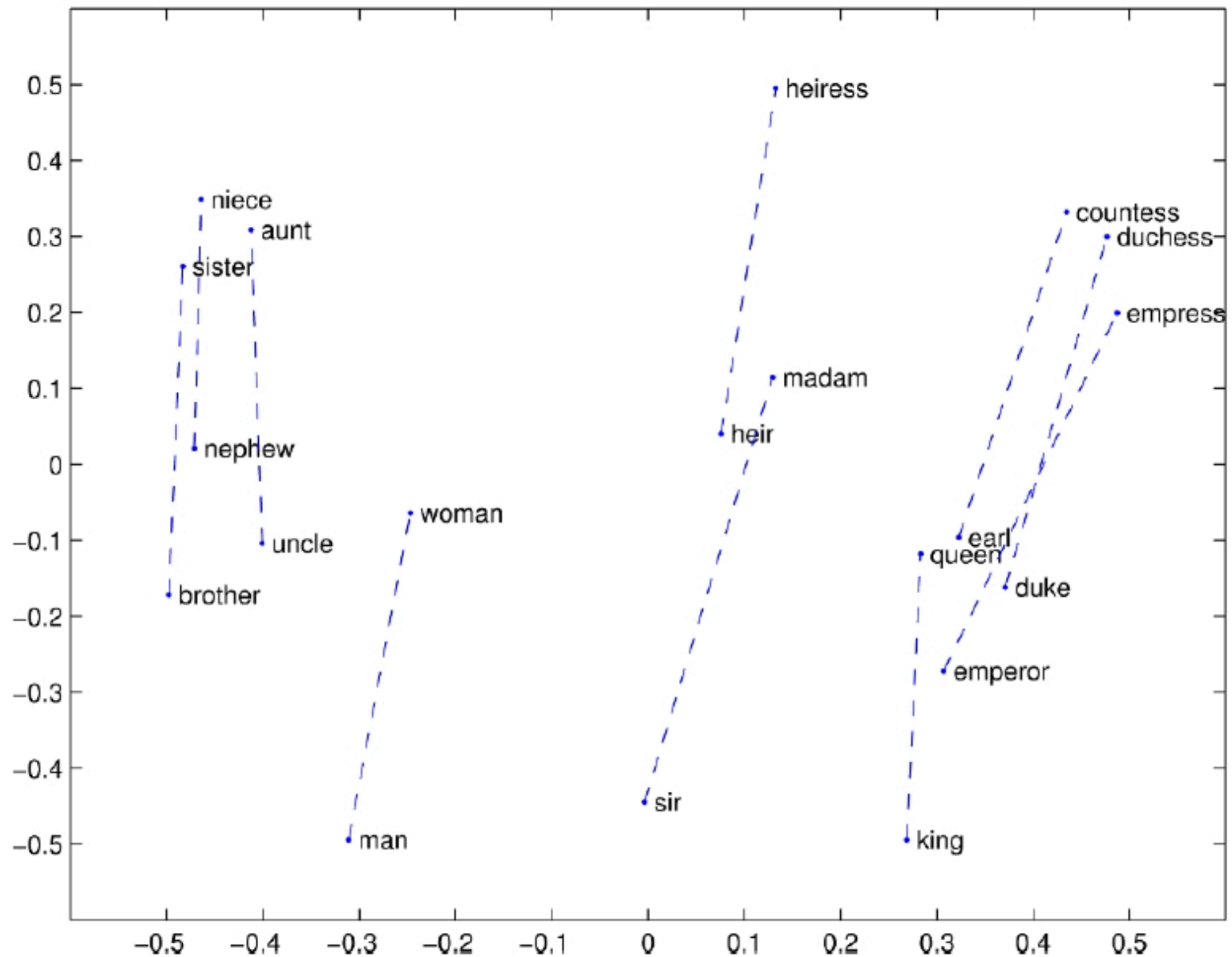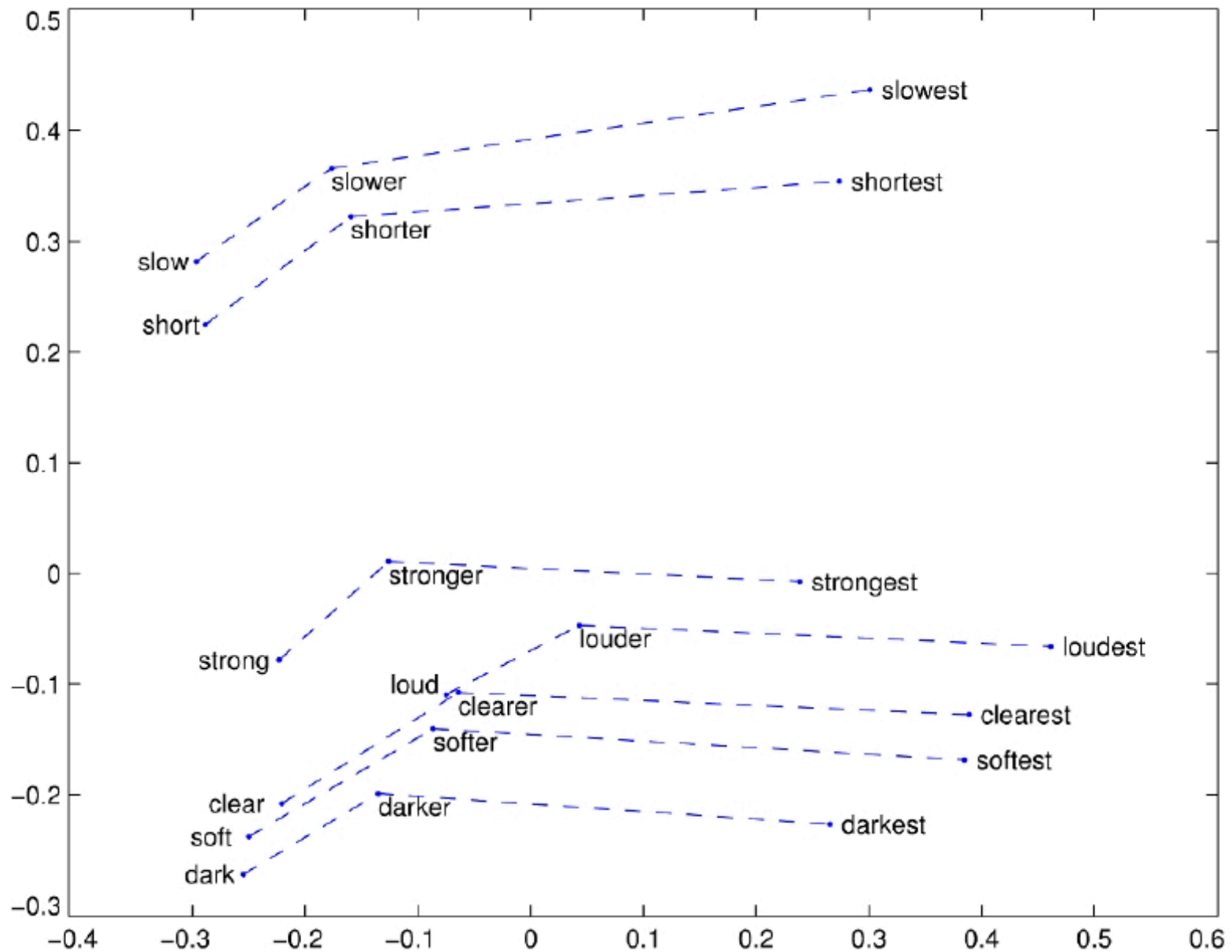
- **historical drift** in semantics (t-SNE projections)



**A**

daft **gay (1900s)**
flaunting · sweet
tasteful · cheerful
pleasant
frolicsome
witty **gay (1950s)**
bright
gays
bisexual
**gay (1990s)** · homosexual
lesbian

**B**

spread
sow
**broadcast (1850s)** seed
sows
circulated · scatter
**broadcast (1900s)**
newspapers
television
radio
bbc **broadcast (1990s)**
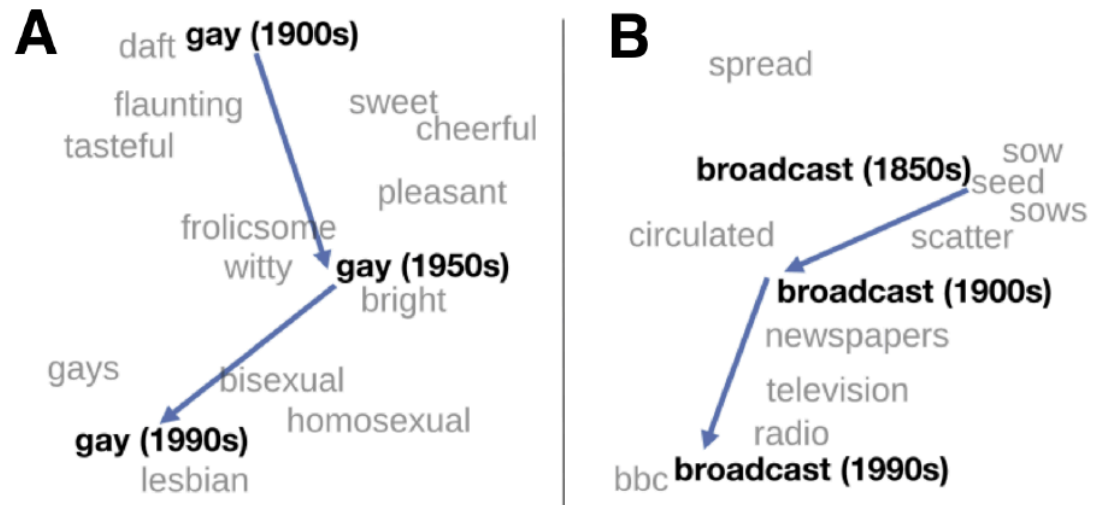
- **gender bias:** father ←→ doctor → mother ←→ nurse

- **racist bias:** Implicit Association Test: USA:
  African American names ←→ unpleasant words;
  European American names ←→ pleasant words
  replicated with GloVe embeddings

# Evaluating Word Vectors

- compare vector similarity against human similarity judgement of raw word pairs (without sentences)

  ○ Word-sim-353 (2002): 353 word pairs, rated on a 0-10 similarity scale

  ○ SimLex-999 (2015)

  ○ TOEFL dataset (1997): 4 proposed synonyms per words: choose closest

- compare vector similarity against human similarity judgement of raw word pairs (with sentences)

  ○ Stanford Contextual Word Similarity (SCWS) (2012): 2003 pairs with sentential context

  ○ semantic textual similarity task (2012)

- analogy task:  a is to b as c is to d; given a, b, and c: find d.
  example: *Athens, Greece, Oslo* → *Norway*
  large datasets exist (2013)
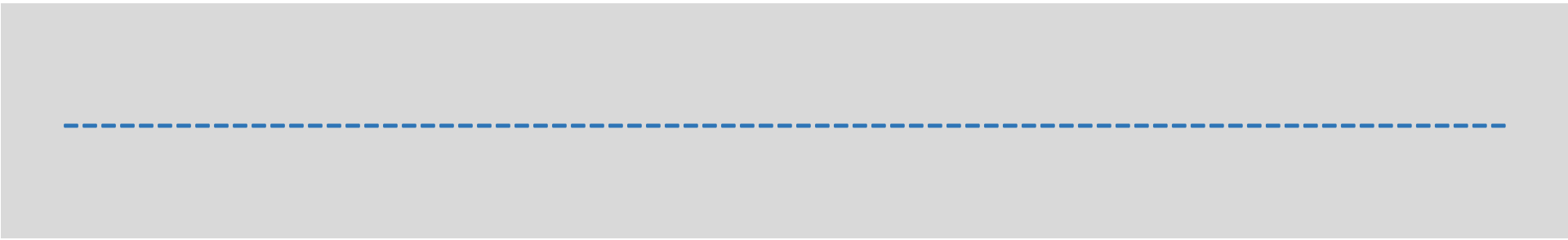
Appendix to this slide set:
Latent Semantic Indexing and
Embeddings via SVD
(see [5])
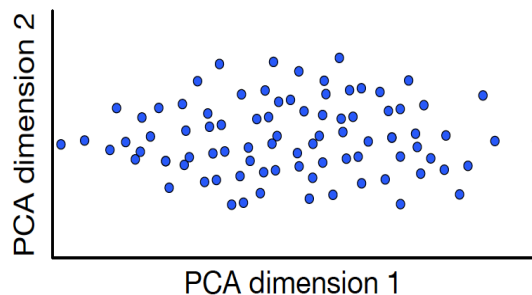
# Repetition from ML1: Principal Component Analysis

- for $N \times D$ pattern matrix $X$ (here e.g. a $|V| \times |V|$ word-word-matrix) compute $D \times D$ covariance matrix $S = \frac{1}{N}(X^T X - \bar{x}\,\bar{x}^T)$

- compute Eigendecomposition of $S$ (e.g. with van Mises power iteration):
  $$S = \Gamma \Lambda \Gamma^T = \sum_{i=1}^{D} \lambda_i \, \gamma_i \gamma_i^T$$
  where $S\gamma_i = \lambda_i \gamma_i$

- restrict $D \times D$ $\Gamma$ to $D \times k$ $\tilde{\Gamma}$ with only the k eigenvectors as columns that correspond k largest eigenvalues $\lambda_i$

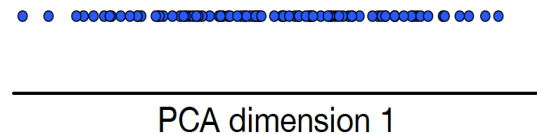- project pattern matrix:
  $X' = X\,\tilde{\Gamma}$
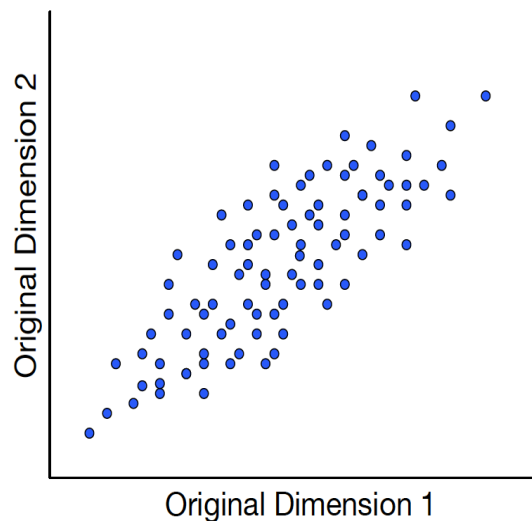


(a)
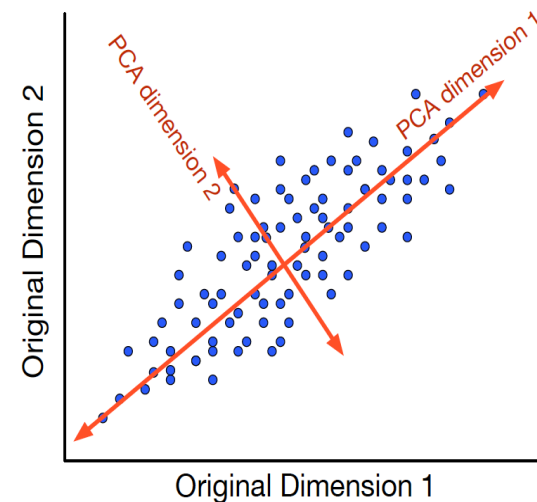
(b)

(c)

(d)

# Repetition from ML1: Singular Value Decomposition

- Decompose $X = U\,\Sigma\,V^T$ with dimensions $N{\times}D = N{\times}N\ \ N{\times}D\ \ D{\times}D$ and

  - $U^T U = I_N$ (columns of $U$ are left singular vectors),

  - $V^T V = V V^T = I_D$ (columns of $V$ are right singular vectors)

  - $\Sigma$ diagonal matrix with at most $\min(N, D)$ many singular values $\sigma_i > 0$ on main diagonal and 0 elsewhere

  - relation to eigen-decomposition: assuming $N \geq D$: $X^T X = V\,\Sigma^T U^T U\,\Sigma\,V^T = V\,\Sigma^T \Sigma\,V^T = V\,diag(\sigma_1^2, \dots \sigma_D^2)\,V^T$, so $X^T X v_i = \sigma_i^2 v_i$ and $\Gamma = V$

  - assuming $N \geq D$: Truncated SVD: restrict

    - $N{\times}N$ matrix $U$ to $N{\times}(N - k)$ matrix $\widetilde{U}$ and

    - $N{\times}D$ diag. matrix $\Sigma$ to $(N - k){\times}(D - k)$ diag. matrix $\widetilde{\Sigma}$ and

    - $D{\times}D$ matrix $V$ to $(D - k){\times}D$ matrix $\widetilde{V}$ and

    dropping the k smallest singular values and their associated singular vectors, effectively producing a low rank approximation $\sum_{i=1}^{D-k} \sigma_i\,u_i v_i^T$ of $X$

  - project $X' = X\,\widetilde{V}$

- example [2]: 7x5 user-movie-rating matrix (rank 3 → columns of $U$ corresponding to zero singular values and respective columns of $\Sigma$ and rows of $V^T$ are dropped <sub>remark due to X having rank = 3 this goes beyond economy-sized SVD here</sub>)

SciFi-concept

Romance-concept

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
=
\begin{bmatrix}
0.13 & -0.02 & -0.01 \\
0.41 & -0.07 & -0.03 \\
0.55 & -0.09 & -0.04 \\
0.68 & -0.11 & -0.05 \\
0.15 & 0.59 & 0.65 \\
0.07 & 0.73 & -0.67 \\
0.07 & 0.29 & 0.32
\end{bmatrix}
\times
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\times
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
-0.12 & 0.02 & -0.12 & 0.69 & 0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Matrix, Alien, Serenity, Casablanca, Amelie

[2]

$X$ — user movie rating matrix

$U$ — user to movie genre similarity matrix

$\Sigma$ — strength of movie genre

$V^T$ — movie genre to movie similarity matrix

possible: drop third column of $U$ and $\Sigma$ and third row of $V^T$ → low rank (rank = 2) approximation of $X$

# Latent Semantic Analysis / Latent Semantic Indexing

Apply SVD to $|V| \times c$ word-document (co-)occurrence matrix X (e.g. using tf-idf or other features). Let rank(X) = m → drop c-m columns of U and rows of $V^T$ corresponding to zero singular values:

$$
\begin{bmatrix} X \end{bmatrix}_{|V| \times c}
=
\begin{bmatrix} W \end{bmatrix}_{|V| \times m}
\begin{bmatrix}
\sigma_1 & 0 & 0 & \dots & 0 \\
0 & \sigma_2 & 0 & \dots & 0 \\
0 & 0 & \sigma_3 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & \sigma_m
\end{bmatrix}_{m \times m}
\begin{bmatrix} C \end{bmatrix}_{m \times c}
$$

SVD notation in Jurafsky:
$$X = W \, \Sigma \, C^T$$

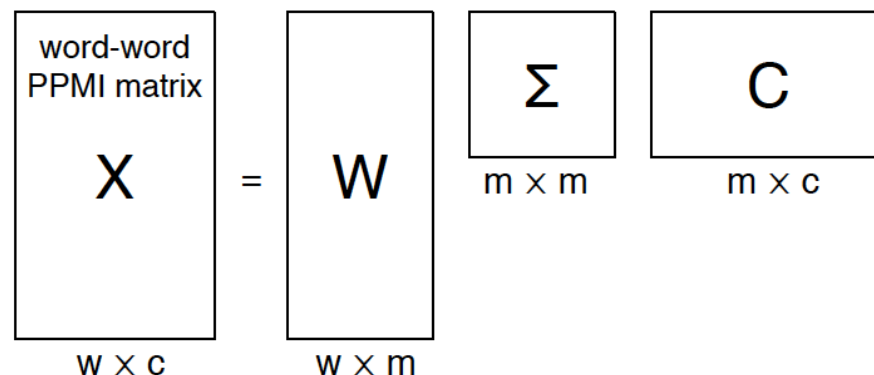Taking only the top $k, k \leq m$ dimensions after the SVD is applied to the co-occurrence matrix X:

$$
\begin{bmatrix} X \end{bmatrix}_{|V| \times c}
=
\begin{bmatrix} W_k \end{bmatrix}_{|V| \times k}
\begin{bmatrix}
\sigma_1 & 0 & 0 & \dots & 0 \\
0 & \sigma_2 & 0 & \dots & 0 \\
0 & 0 & \sigma_3 & \dots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \dots & \sigma_k
\end{bmatrix}_{k \times k}
\begin{bmatrix} C \end{bmatrix}_{k \times c}
$$

k or m many document genres / document classes

# SVD on Word-Word-Co-Occurrence Matrices

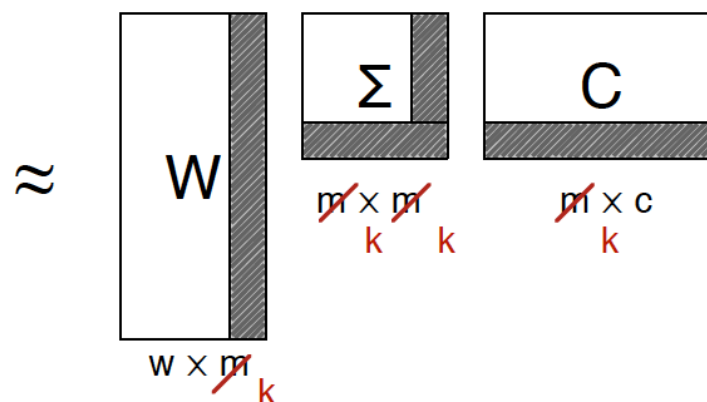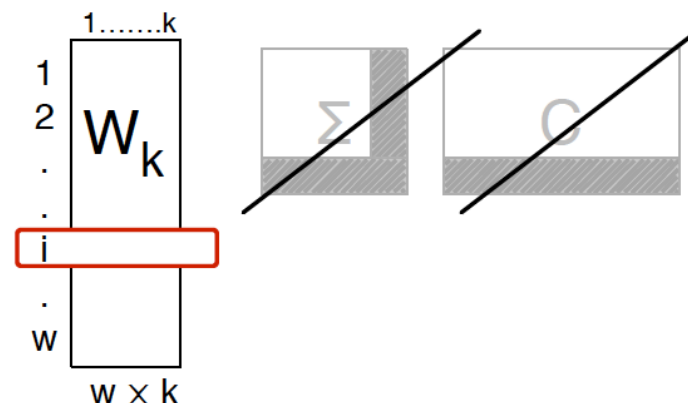- SVD of a PPMI-weighted word-word matrix

- → c is larger than for word-doc case → use k = 500...5000

remark: for some applications the less "compact" more noisy word-vectors / embeddings (rows) of the raw PPMI matrix X may even perform better

1) SVD

word-word PPMI matrix
$X$
$w \times c$
$=$
$W$
$w \times m$
$\Sigma$
$m \times m$
$C$
$m \times c$

2) Truncation:

$\approx$
$W$
$w \times \cancel{m}\,k$
$\Sigma$
$\cancel{m} k \times \cancel{m} k$
$C$
$\cancel{m} k \times c$

3) Embeddings:

$1.......k$
$1$
$2$
.
.
$i$
.
$w$
$W_k$
$w \times k$

embedding for word i:

$\Sigma$    $C$

here is a formula
can you explain this formula advantage and disadvantage from laplace

sorular boyle olacakmis

formülü veririm
kneeser ney neyde neyde iyi

sonra da understanding rememberingden geçer dedi asdşkasşd
yani anlaman iççin zaten formülü bilmem lazım

# Bibliography

(1)  Dan Jurafsky and James Martin: Speech and Language Processing (3$^{rd}$ ed. draft, version Jan 2022); Online: https://web.stanford.edu/~jurafsky/slp3/ (URL, Oct 2022) (this slide set is especially based on chapter6)

(2)  Günnemann, Groh, Bojchevski, Shchur: Machine Learning 1, TUM-IN2064, WS 2017/18: Lecture + Tutorial Material on Dimensionality Reduction (2018)

(3)  Dan Jurafsky and James Martin: Speech and Language Processing (3$^{rd}$ ed. draft, version Oct. 2019); (URL, Oct 2019): chapter 19

(4)  Dan Jurafsky and James Martin: Speech and Language Processing (3$^{rd}$ ed. draft, version Aug 2017), section 16.1. "Dense Vectors with SVD"

(5)  Murphy: Machine Learning, MIT Press (2012), Chapter 12, esp. Section 12.2.3

(6)  Mikolov et al.: "Distributed Representations of Words and Phrases and their Compositionality." In NIPS2013 , pp. 3111--3119. 2013

# Recommendations for Studying

- **minimal approach:**

  work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- **standard approach:**

  minimal approach + read the corresponding pages in Jurafsky [1]

- **interested students**

  == standard approach + study the slides 37ff