Vector semantics

ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

25/03 - Gustave Cortal

Lemmas and senses

lemma

sense

mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

A sense is the meaning component of a word

Lemmas can be **polysemous** (have multiple senses)

Synonyms

Synonyms have the same meaning in some or all contexts

couch / sofa
big / large
automobile / car
water / H_20

Similarity

Words sharing elements of meaning

car / bicycle

cow / horse

french / english

Word association (relatedness)

Words can be related in any way, perhaps via a semantic frame or field

coffee / tea: **similar**

coffee / cup: **related**

Semantic field

Words that cover a particular semantic domain

hospitals

surgeon, *scalpel*, *nurse*, *anaesthetic*, *hospital*

restaurants

waiter, *menu*, *plate*, *food*, *menu, chef*

houses

door, *roof, kitchen*, *family*, *bed*

Antonyms

Senses that are opposites with respect to only one feature of meaning

Otherwise, they are very similar

dark / light   short / long   fast / slow   rise / fall

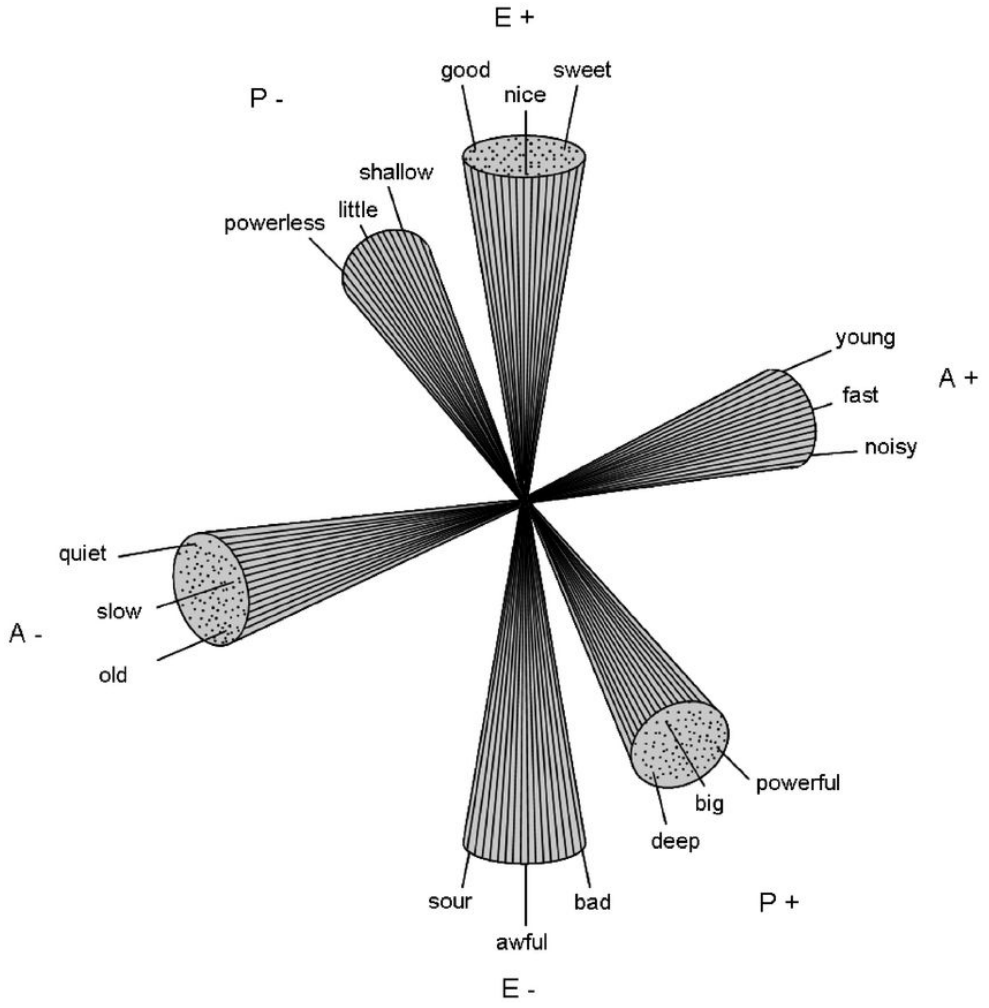
hot / cold   up / down   in / out

Connotation

Words have **affective** meanings

Positive (*happy*) and negative connotations (*sad*)

A three-dimensional affective space of connotative meaning by Osgood et al. (1957)



Can we build a theory of how to represent word meaning ?

Ludwig Wittgenstein

“The meaning of a word is its use in the language”

Define words by their usages

Words are defined by the words around them (the environment)

Zellig Harris (1954): if A and B have almost identical environments, then they are synonyms

Idea 1: defining meaning by linguistic distribution

Idea 2: meaning as a point in multidimensional space

Defining meaning as a point in space based on distribution

Each word is a vector

Similar words are **nearby in the semantic space**

We build this space automatically by seeing which words are nearby in text

Meaning of a word as a vector

Called an “embedding” because it's embedded into a vector space

Recent NLP models use embeddings as the representation word meaning

Two main kinds of embeddings

tf-idf

- **Sparse** vectors
- Words are represented by the **counts** of nearby words

Word2vec

- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
- Later we'll discuss extensions called **contextual embeddings**

Term frequency - inverse document frequency (tf-idf)

Term-document matrix: word 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

battle is “the kind of word that occurs in Julius Caesar and Henry V”

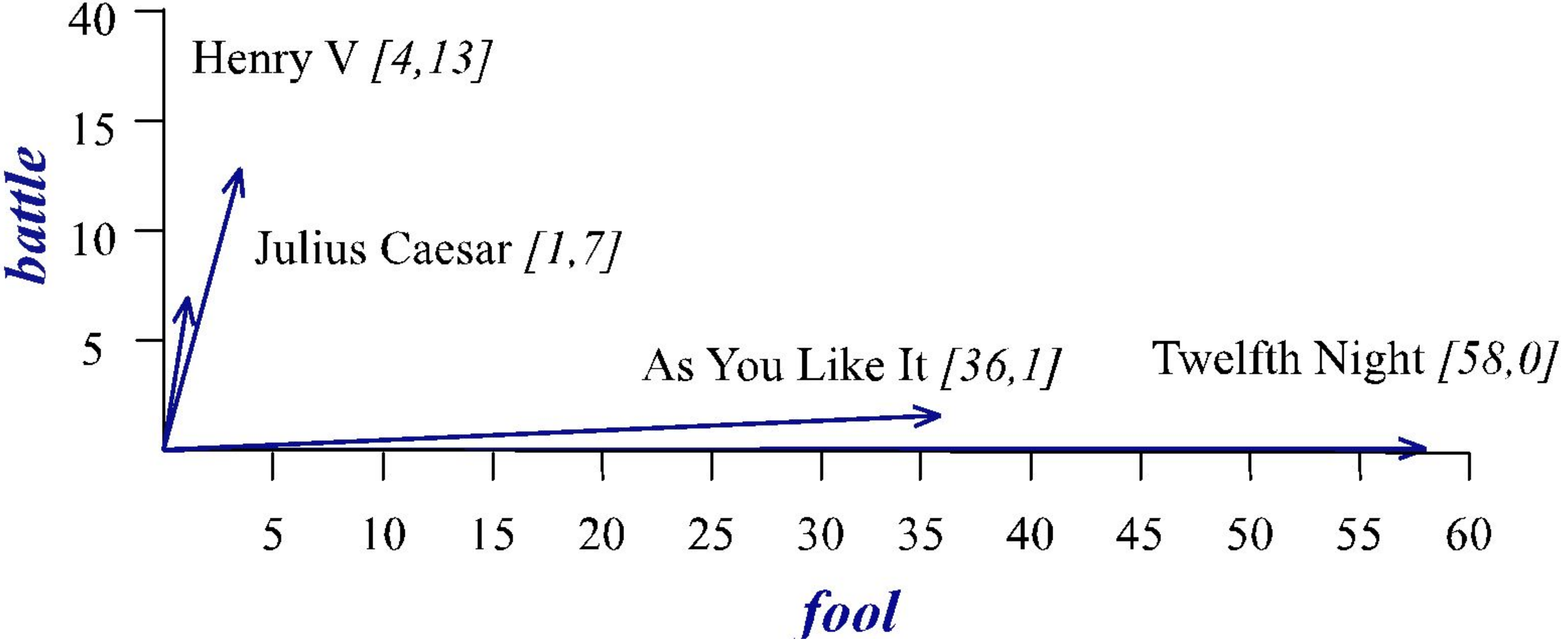
fool is “the kind of word that occurs in As You Like It and Twelfth Night”

Term-document matrix: document vectors

Each document is represented by a vector of words

	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

Visualizing document vectors

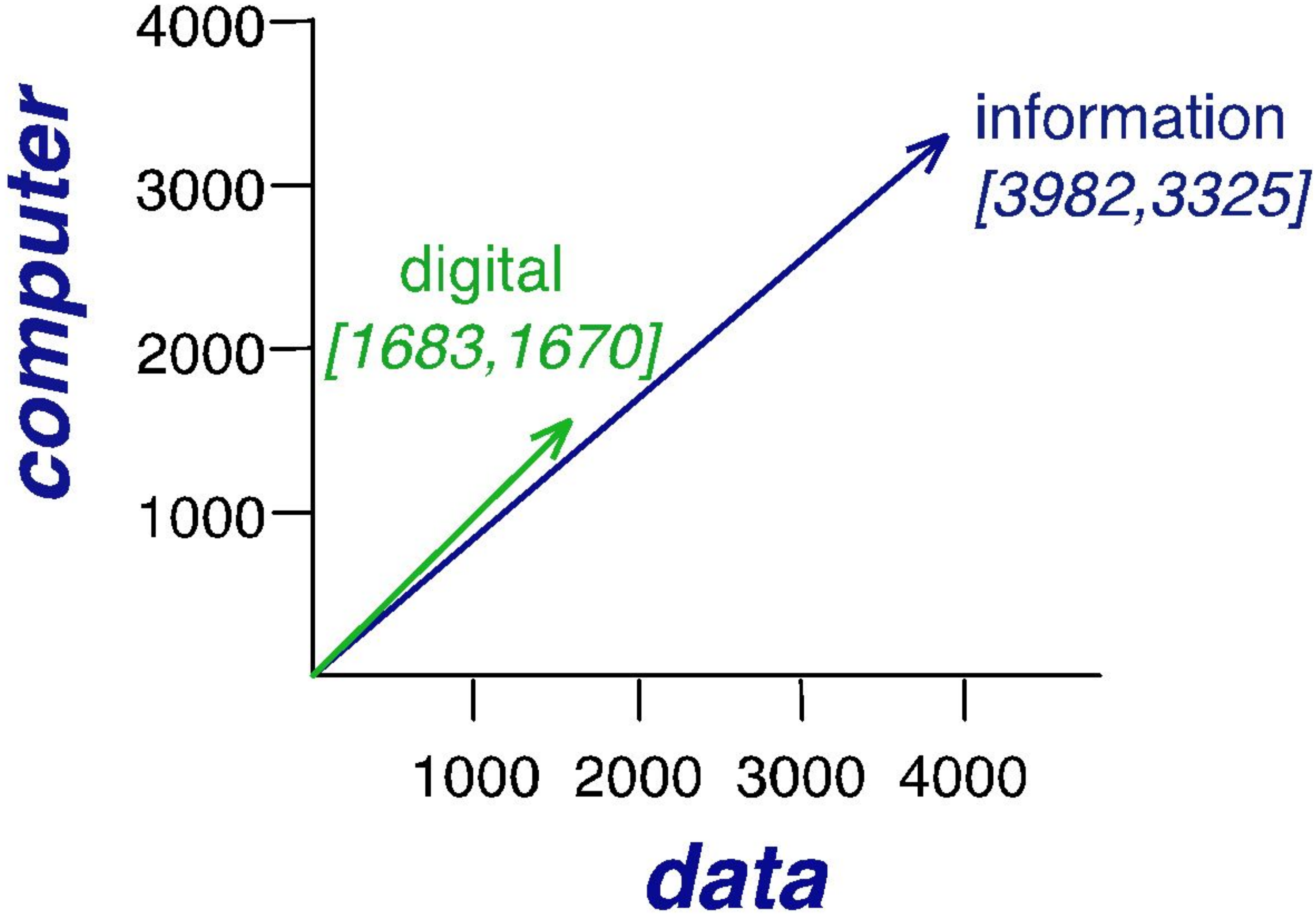


Word-word matrix or "term-context matrix"

Two words are similar in meaning if their context vectors are similar

is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

	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	...



4000
3000
2000
1000
computer
information
[3982,3325]
digital
[1683,1670]
1000
2000
3000
4000
data

Distance metrics for computing word similarity

Dot product as a similarity metric

The dot product between two vectors is a scalar:

$$\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

The dot product tends to be high when the two vectors have large values in the same dimensions

Problem with dot-product

Dot product is higher if a vector is longer → it favors long vector

Frequent words (*of*, *the*, *you*) have long vectors since they occur many times with other words

→ dot product favors frequent words

Vector length:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Alternative: cosine as a similarity metric

$$\text{cosine}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal

Since raw frequency values are non-negative, the cosine for term-term matrix vectors ranges from 0 to 1

Cosine examples

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

Raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information
- But overly frequent words like *the*, *it,* or *they* are not very informative about the context

Term frequency (tf) in the tf-idf algorithm

We could imagine using raw count:

$$\text{tf}_{t,d} = \text{count}(t,d)$$

But instead of using raw count, we usually squash a bit:

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Document frequency (df)

df_t is the number of documents *t* occurs in

Romeo is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\mathrm{idf}_t = \log_{10}\left(\frac{N}{\mathrm{df}_t}\right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

What is a document?

Could be a tweet, a Wikipedia article, etc.

Documents can be **anything**

Final tf-idf weighted value for a word

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

Raw counts:

	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

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.246	0	0.454	0.520
good	0	0	0	0
fool	0.030	0.033	0.0012	0.0019
wit	0.085	0.081	0.048	0.054

Word2vec: learning the embeddings

Sparse versus dense vectors

tf-idf vectors are

- **long** (length |V|= 20,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors which are

- **short** (length 50-1000)
- **dense** (most elements are non-zero)

Sparse versus dense vectors

Why dense vectors?

- Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)

- Dense vectors may **generalize** better than explicit counts

Word2vec

Instead of **counting** how often each word *w* occurs near "*apricot*"

- Train a classifier on a **prediction task**:
  - Is *w* likely to show up near "*apricot*" *and* "*epita*"?

We don't actually care about this task

- But we'll take the learned classifier weights as the word embeddings

→ **Self-supervision** (no need for human labels)

Approach: predict if candidate word *c* is a “neighbor”

1. Treat the target word *t* and a neighboring context word *c* as **positive examples**
2. Randomly sample other words in the vocabulary to get negative examples (optional)
3. Train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Properties of embeddings

The kinds of neighbors depend on window size

Small windows (C= +/- 2) : nearest words are syntactically similar words

- *Hogwarts* nearest neighbors are other fictional schools
  - *Sunnydale, Evernight, Blandings*

Large windows (C= +/- 5) : nearest words are related words in same semantic field

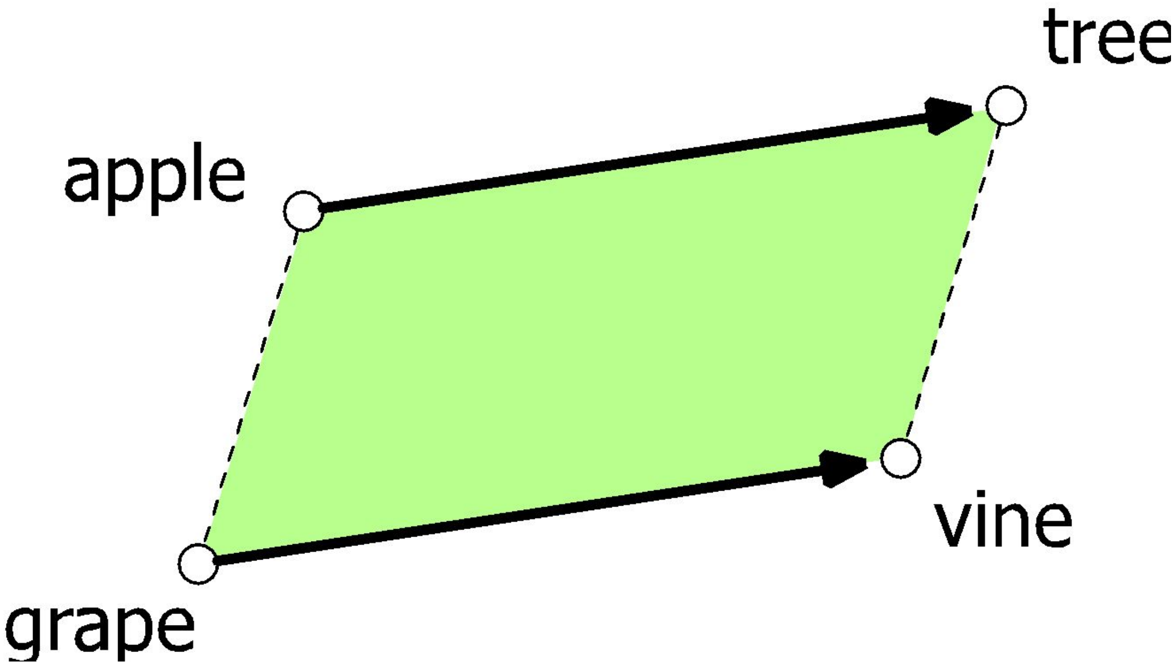
- *Hogwarts* nearest neighbors are Harry Potter world:
  - *Dumbledore, half-blood, Malfoy*

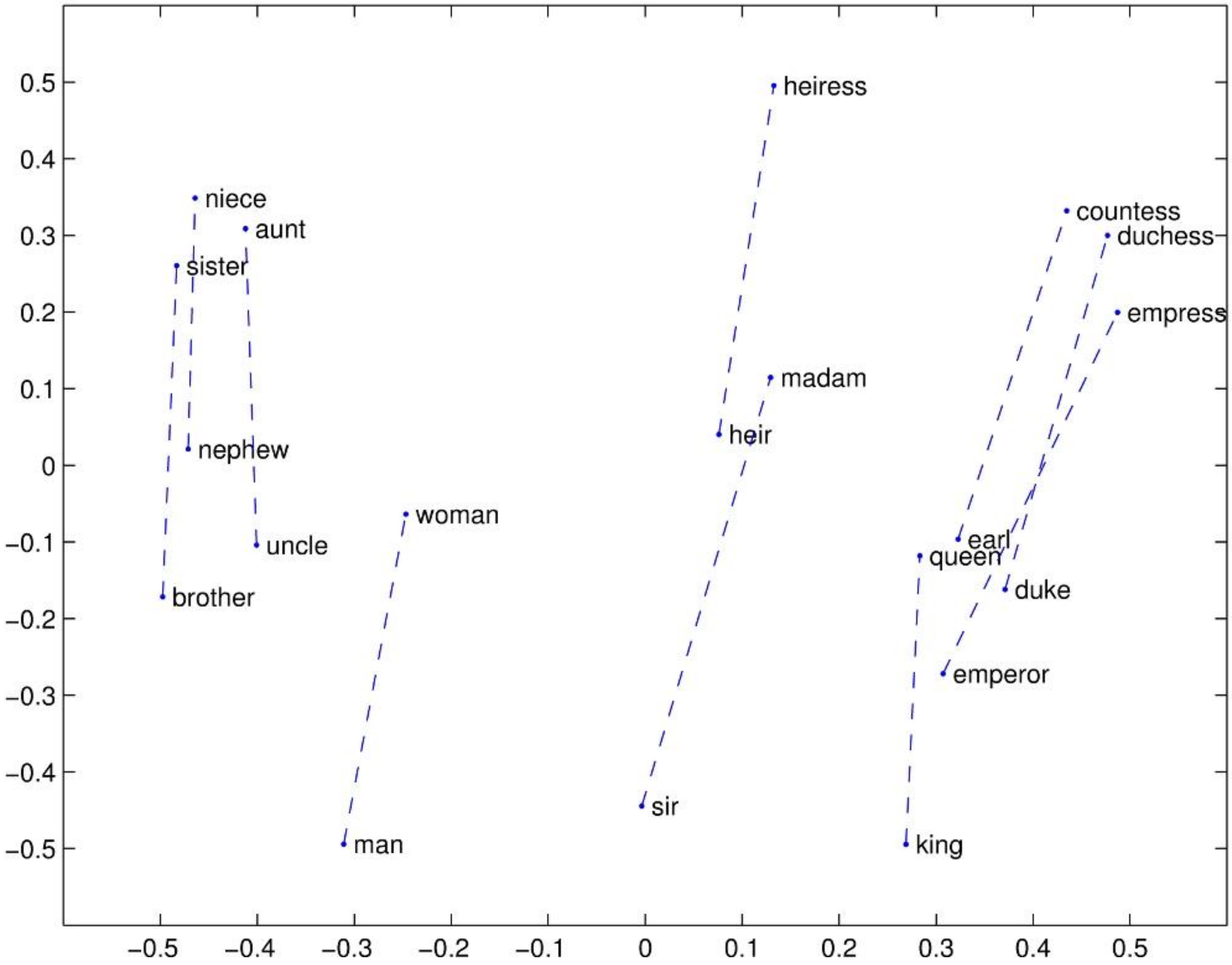
Analogical relations

The parallelogram model of analogical reasoning

To solve: *"apple is to tree as grape is to _____"*

Add <u>tree</u> – <u>apple</u> to <u>grape</u> to get <u>vine</u>





heiress
niece
aunt
sister
countess
duchess
empress
madam
heir
nephew
woman
uncle
earl
queen
duke
brother
emperor
sir
man
king
0.5
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−0.5
−0.5
−0.4
−0.3
−0.2
−0.1
0
0.1
0.2
0.3
0.4
0.5

Embeddings as a window onto historical semantics

Train embeddings on different decades of historical text to see meanings shift

~30 million books, 1850-1990, Google Books data

William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.