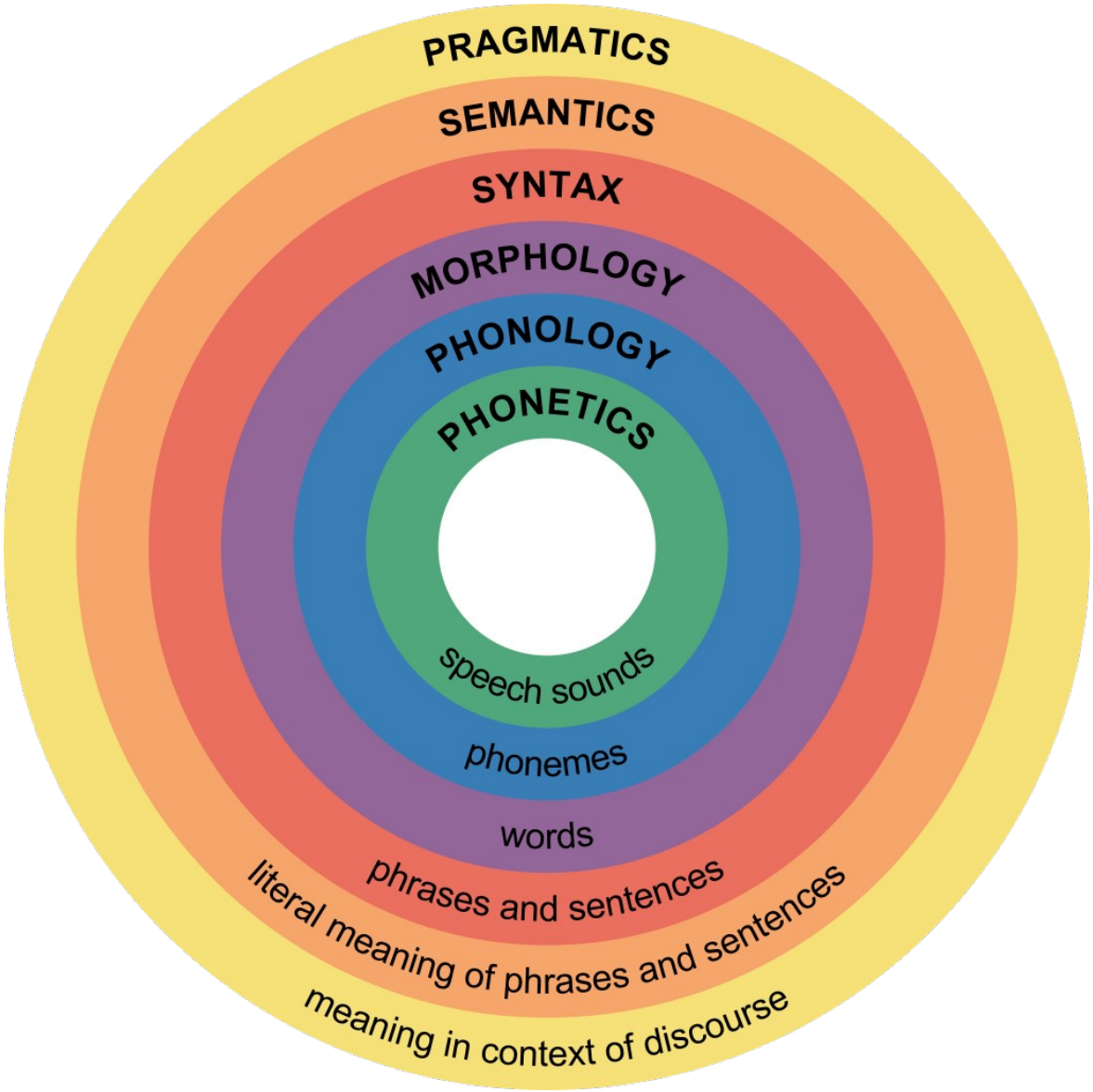
Introduction to Natural Language Processing
Text preprocessing & word representations

Ana Sabina Uban
auban@fmi.unibuc.ro
https://nlp.unibuc.ro/master_en.html

+ some slides credits: Dan Jurafsky, James Martin, Chris Manning, Reda Bouadjenek, Pandu Nayak and Prabhakar Raghavan, Liviu Dinu

Levels of linguistic analysis

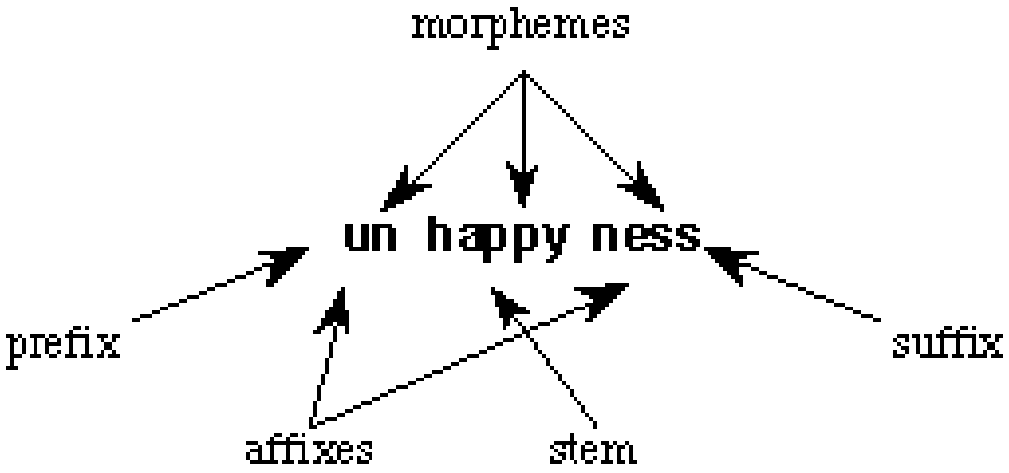


Levels of linguistic analysis

Phonetic or phonological level: deals with pronunciation

Levels of linguistic analysis

Morphological level: deals with the smallest parts of words that carry meaning, and suffixes and prefixes.

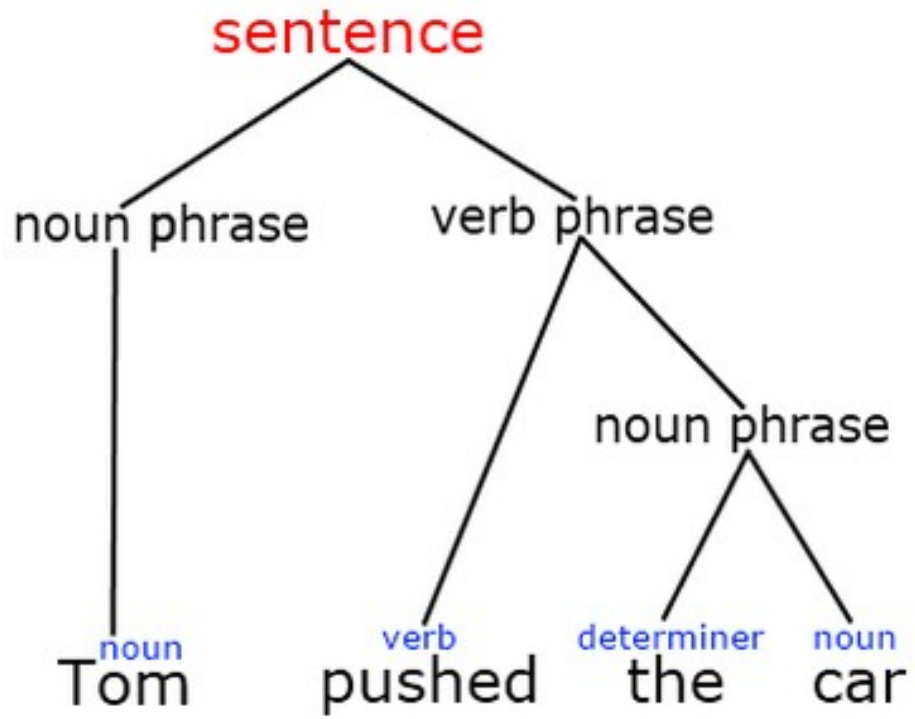


Levels of linguistic analysis

Lexical level: deals with words.

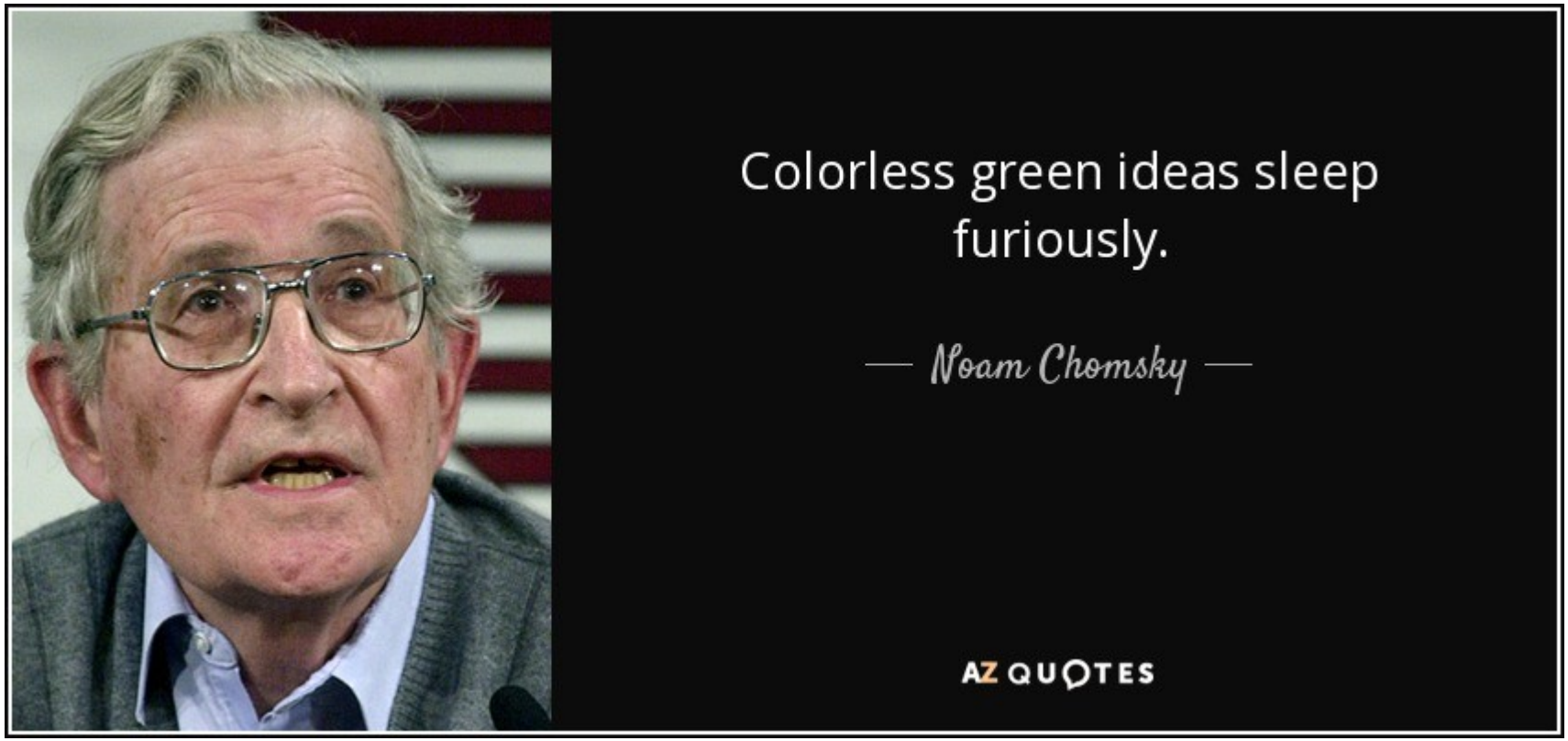
Levels of linguistic analysis

Syntactic level: deals with grammar and structure of sentences.



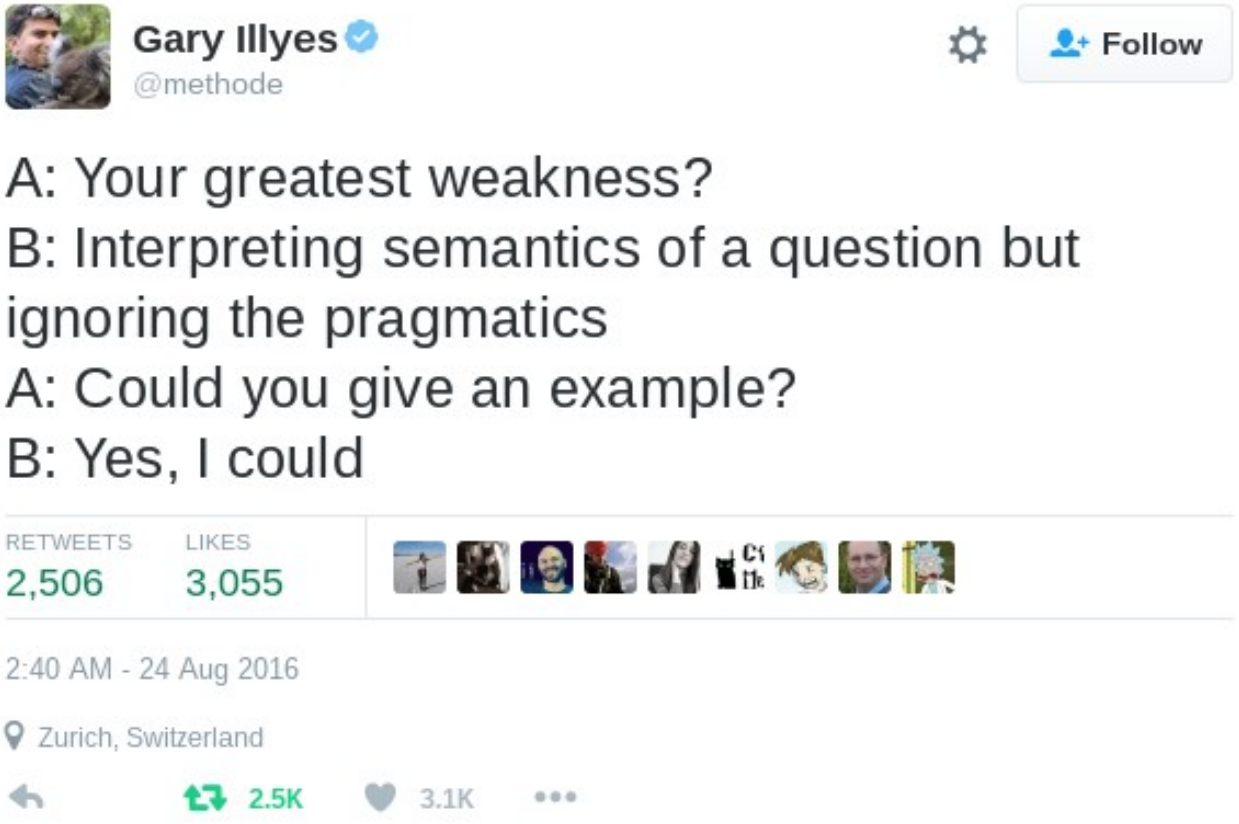
Levels of linguistic analysis

Semantic level: deals with the meaning of words and sentences.



Levels of linguistic analysis

Pragmatic level: deals with the knowledge that comes from the outside world, i.e., from outside the content of the document.



Natural Language Processing

Natural language processing *= the application of computational techniques to the analysis and synthesis of natural language and speech*

- Large variety of different tasks, useful as standalone applications or as steps in a more complex NLP pipeline
- Solutions can be algorithm/rule-based, can be trained classifiers (machine learning), unsupervised information extraction etc

Examples of NLP tasks

- Syntactic parsing (input: text, output: syntactic structure)
- Sequence labelling: POS (part-of-speech) tagging, NER (named entity recognition) (input: text, output: labels for each token)
- Text classification (sentiment analysis, fake news detection, offensive language detection etc)
- Information retrieval (input: query, output: ranked documents)
- Topic modelling (input: text, output: detected topics)
- Language modelling & text generation: summarization, ChatGPT, machine translation (input: text?, output: text)

What do words mean?

Lexical semantics:
the study of word meaning

First: What is a word, and how do we extract and represent it numerically?

Processing text data

Preprocessing & Converting texts to numbers

Main steps

- Machine learning models need numerical inputs, so preprocessing is basically the process of taking a raw chunk of text and converting it into numbers.

- Therefore, for most applications, preprocessing can roughly be divided into the following 3 steps:

  - **Step 1: Normalization (Cleaning)**
  - **Step 2: Segmentation (Tokenization) => words!**
  - **Step 3: Numericalization (Vocabulary mapping)**

Definitions

- **Normalization**: is where we clean the data in advance to remove unwanted inputs and to convert certain characters/sequences into canonical forms.
- **Tokenization** is breaking a text chunk in smaller parts. Whether it is breaking paragraphs into sentences, sentences into words or words into characters.
- **Numericalization**: is where we convert the textual entities into numbers/ids so that we can feed them to our model.

Main difficulty:

- Language dependent!
- Domain dependent
- Application dependent

Normalization (Cleaning)

- Refers to the process of cleaning up the input and mapping characters/words to a "canonical" form.
- **Standard steps:**
  - Handling repeating characters (e.g. "goooooood" -> "good")
  - Handling homoglyphs (e.g. "$tupid" -> "stupid")
  - Mapping special inputs such as URLs, email addresses, and HTML tags to a canonical form (e.g. "http://www.foo.com/bar" -> "[URL]")
  - Unicode normalization

Normalization (Cleaning)

- Need to “normalize” terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***

- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
  - e.g.: mapping all characters to their lowercase form (Ciao=ciao).

Normalization (Cleaning)

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
- For sentiment analysis, machine translation, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Segmentation / Tokenization

- Segmentation/Tokenization is probably the most complex part of the preprocessing pipeline.
  - Naive tokenization algorithms,
  - Rule-based tokenizers,
  - modern (subword) tokenizers that are learned on data.

Tokenizing on Whitespace/Punctuation

- The most naive form of tokenization.
- "I saw a girl with a telescope."
  - This would be split into
- "I", "saw", "a", "girl", "with", "a", "telescope.“
- **Never** use purely whitespace tokenization in NLP!
- A slightly better approach is to tokenize based on punctuation like
  - "I", "saw", "a", "girl", "with", "a", "telescope", "."

Sentence Segmentation

- **What is a sentence?**
- The first answer to what is a sentence is something ending with a “.” or “!”
- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a “.”
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

Lemmatization, morphology?

- Reduce inflections or variant forms to base form (a type of normalization)
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
- *the boy's cars are different colors → the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation

Why?

- Many NLP tasks can benefit from lemmatization.
- Examples:
  - Topic modelling looks at word distribution in a document.
  - By normalizing words to a common form, we can get better results in ML models (In word embeddings removing inflected wordforms can improve downstream NLP tasks.)
  - For information retrieval (IR), lemmatization helps with query expansion so that suitable matches are returned even if there's not an exact word match.
  - In document clustering, it's useful to reduce the number of tokens. It also helps in machine translation.
- The decision to use lemmas is application dependent; we should use lemmas only if they show better performance.

Lemmatization. How?

- Use predefined dictionaries:
  - https://cst.dk/download/cstlemma/
  - https://www.clarin.si/repository/xmlui/handle/11356/1041
  - WordNet (accessed directly through python libraries: NLTK, Spacy - see lab)

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Stemming

adjustable → adjust

formality → formaliti

formaliti → formal

airliner → airlin ⚠

Lemmatization

was → (to) be

better → good

meeting → meeting

Porter's algorithm. The most common English stemmer

Step 1a

```
sses → ss        caresses → caress
ies  → i         ponies   → poni
ss   → ss        caress   → caress
s    → Ø     cats       → cat
```

Step 1b

```
(*v*)ing → Ø  walking   → walk
              sing      → sing
(*v*)ed  → Ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize        digitizer → digitize
ator→ ate        operator  → operate
…
```

Step 3 (for longer stems)

```
al    → Ø   revival    → reviv
able  → Ø   adjustable → adjust
ate   → ø  activate   → activ
…
```

Romanian Snowball

https://snowballstem.org/algorithms/romanian/stemmer.html

- ocol
  ocolea
  ocolesc
  ocoleşte
  ocoleşti
  ocoli
  ocolim
  ocolind
  ocolire
  ocolişuri
  ocolit
  ocolită
  ocoliţi
  ocolul
  ocoluri
  ocolurile

- ocol
  ocol
  ocol
  ocol
  ocol
  ocol
  ocol
  ocol
  ocol
  ocolişur
  ocol
  ocol
  ocol
  ocol
  ocolur
  ocolur

Choosing Normalization Steps

- What kinds of normalization should we actually apply?
  - No clear answers to this, but:
- Always look at the data manually!
- Are you using a pretrained model (e.g. BERT)?
  - If so, make sure your preprocessing steps match the preprocessing that is conducted for pretraining. This can be more difficult than it seems, since papers don't always report all their preprocessing steps.

In the end, depends on the data, the application and the

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)
- In ML applications: we usually build a vocabulary of types after normalization, and keep the most frequent ones (discard very rare terms)

How many words?

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Practical examples using Python

Code along!

https://colab.research.google.com/drive/1UosXW-s-yc-NBbaD5cqE-uq4jvvlFdtN?usp=sharing

Pre-processing for NLP

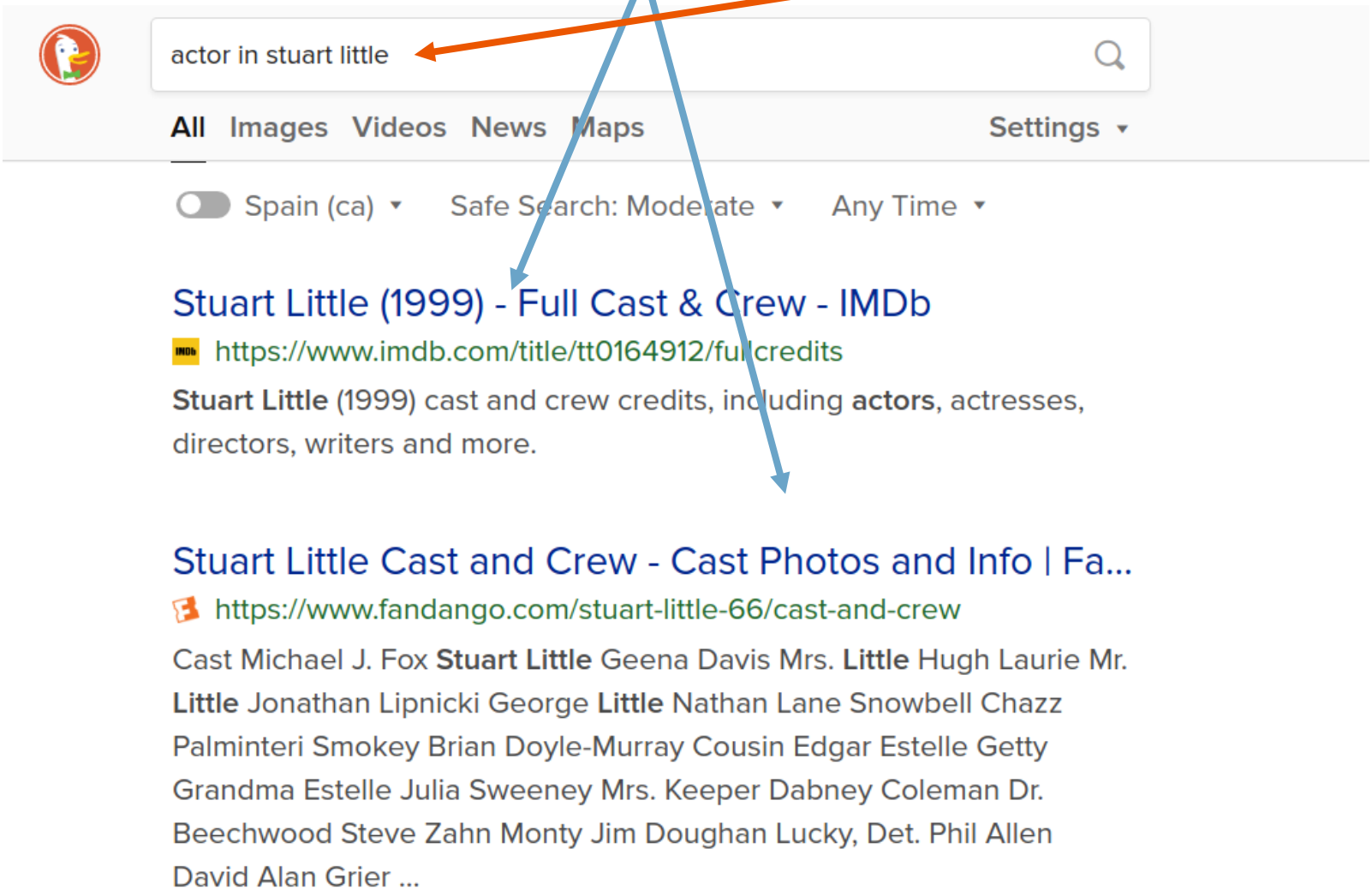
Normalization: is where we clean the data in advance to remove unwanted inputs and to convert certain characters/sequences into canonical forms.

Tokenization *is breaking a text chunk in smaller parts. Whether it is breaking Paragraph in sentences, sentence into words or word in characters.*

Numericalization*:* is where we convert the textual entities into numbers/ids so that we can feed them to our model.

Information Retrieval:

finding relevant documents wrt a query



Scoring as the basis of ranked retrieval

- We wish to return in order the documents most likely to be useful to the searcher
- How can we rank-order the documents in the collection with respect to a query?
- Assign a score – say in [0, 1] – to each document
- This score measures how well document and query "match".

Query-document matching scores

- We need a way of assigning a score to a query/document pair
- Let's start with a one-term query
- If the query term does not occur in the document: score should be 0
- The more frequent the query term in the document, the higher the score (should be)
- We will look at a number of alternatives for this.

Binary term-document incidence matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Each document is represented by a binary vector $\in \{0,1\}^{|V|}$

Term-document count matrices

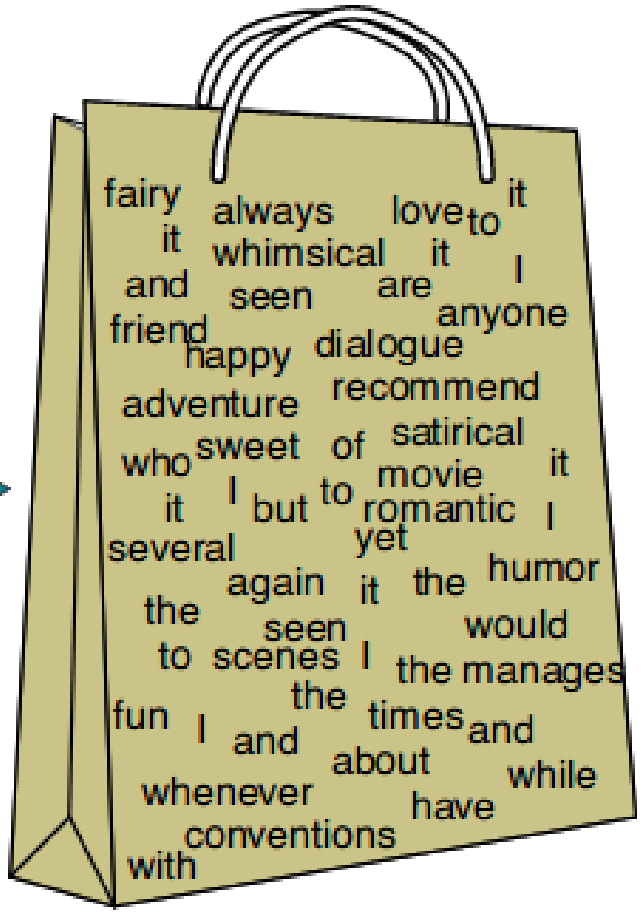
- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^{|V|}$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation doesn't consider the ordering of words in a document
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors
- This is called the bag of words model.

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Source: https://dudeperf3ct.github.io/lstm/gru/nlp/2019/01/28/Force-of-LSTM-and-GRU/

Term-document matrix

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

Vectors are the basis of information retrieval

	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

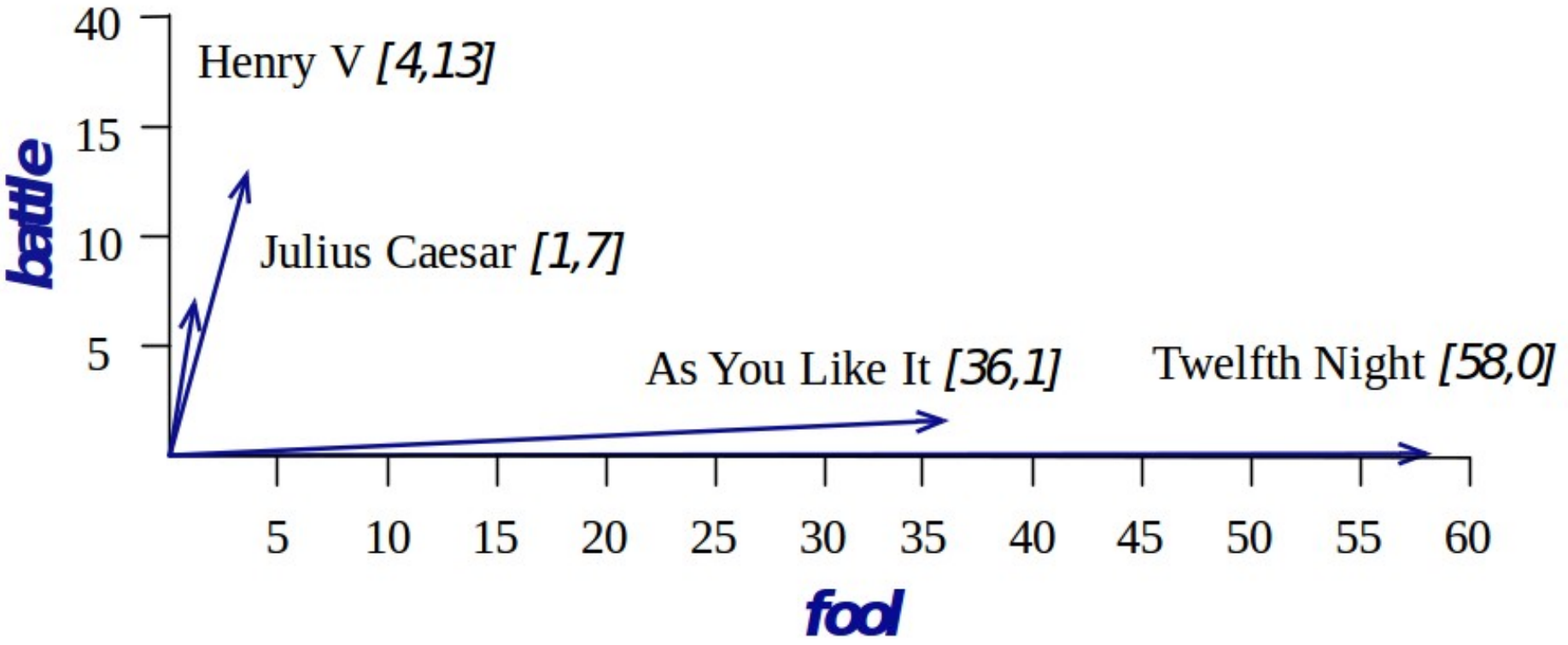
Vectors are similar for the two comedies

Comedies have more fools and wit and fewer battles.

Documents as vectors

- So we have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
- These are very sparse vectors - most entries are zero.

Visualizing document vectors

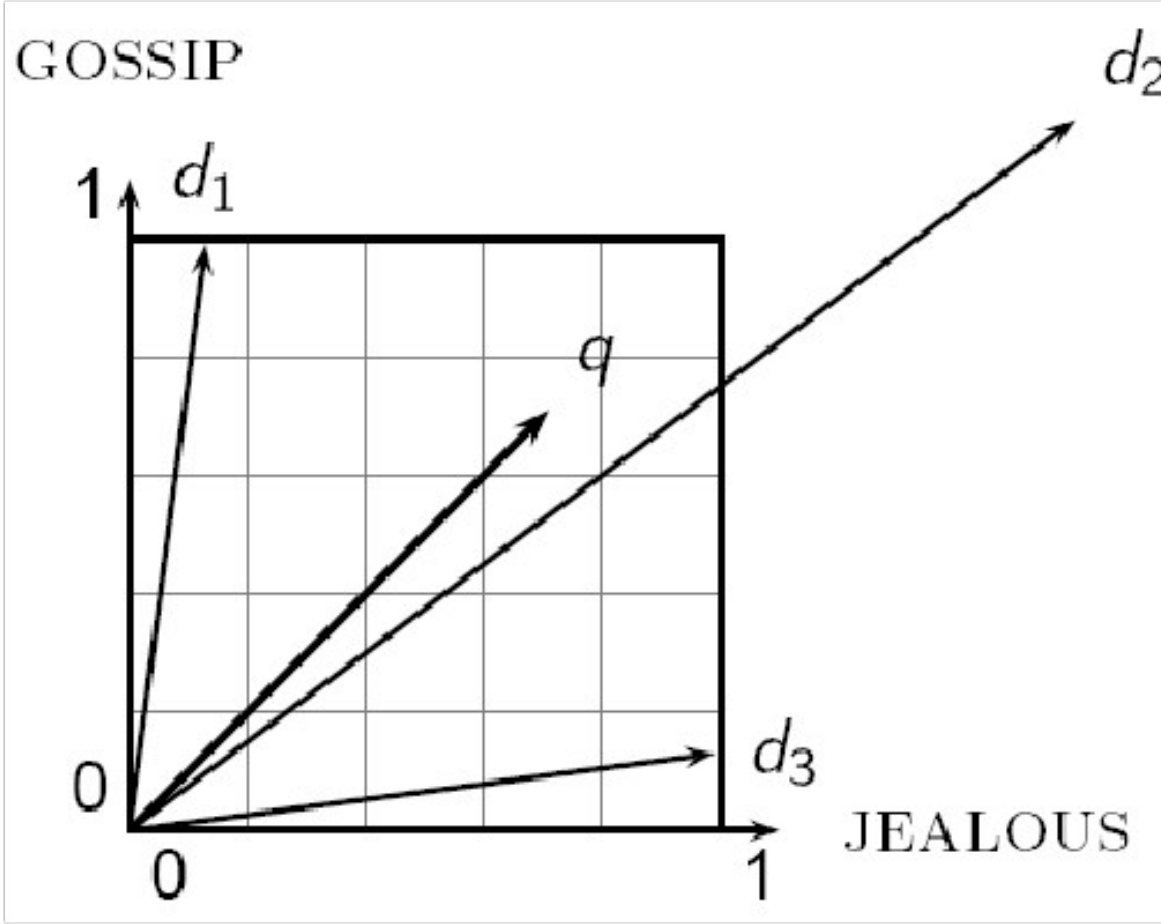


Formalizing vector space proximity

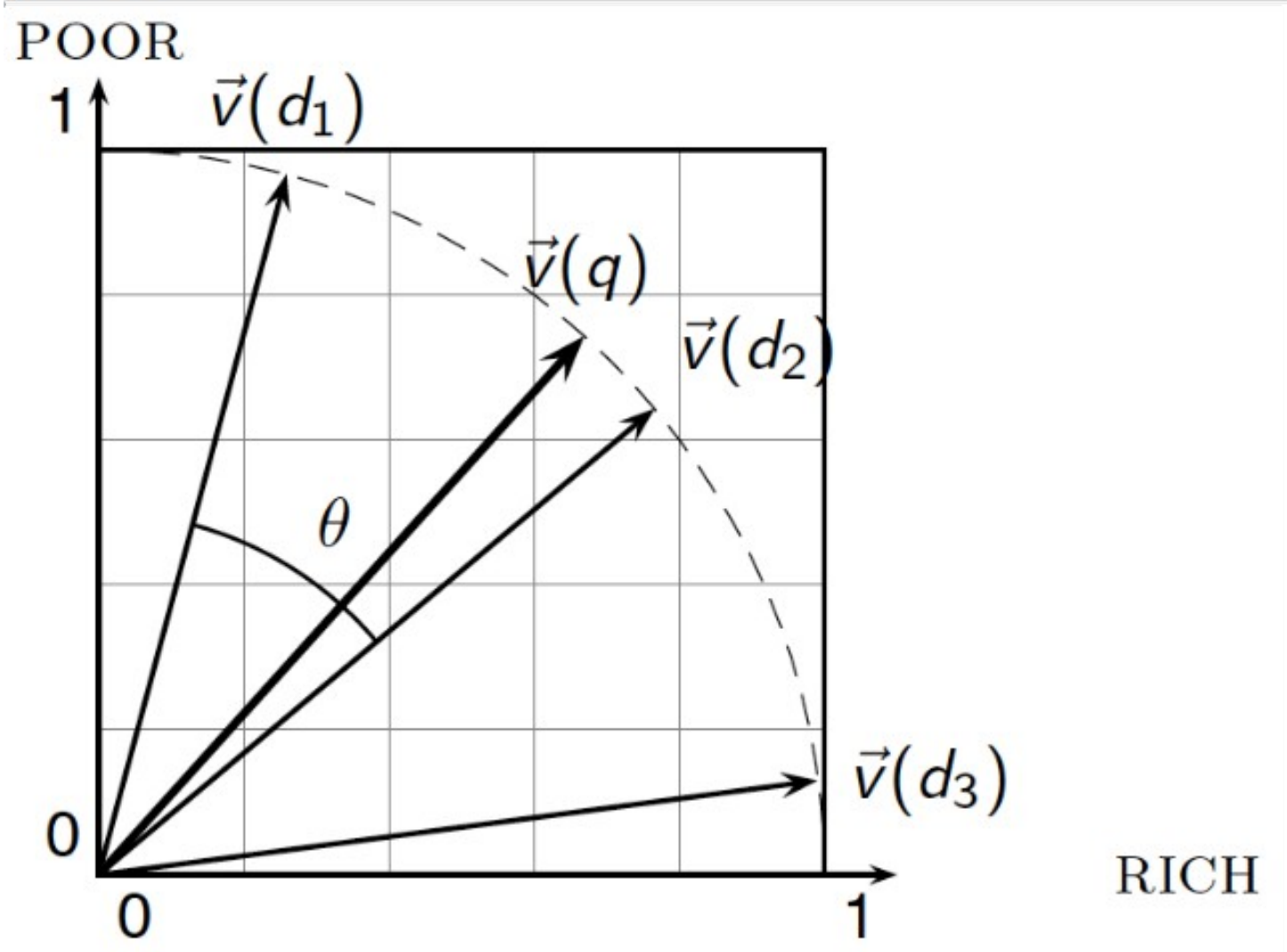
- First cut: distance between two points
  - (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why distance is a bad idea

The Euclidean distance between $\vec{q}$ and $\vec{d_2}$ is large even though the distribution of terms in the query $\vec{q}$ and the distribution of terms in the document $\vec{d_2}$ are very similar.



Cosine similarity illustrated



Use angle instead of distance

- Thought experiment: take a document *d* and append it to itself. Call this document *d′*.
- "Semantically" d and d′ have the same content
- The Euclidean distance between the two documents can be quite large
- The angle between the two documents is 0, corresponding to maximal similarity.

- Key idea: Rank documents according to angle with query.

Cosine for computing similarity

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

v_i is the count for word i in document v

w_i is the count for word i in document w.

Cos(v,w) is the cosine similarity of v and w

But raw frequency is a bad representation

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

Need a function that resolves this frequency paradox!

Query: the movie that I liked the most

Document1: The cat is a domestic species of small carnivorous mammal. It is the only domesticated species in the family Felidae and is often referred to as the ...

Document 2: Stuart Little: best movie ever!

Document frequency

- Rare terms are more informative than frequent terms
  - Recall stop words
- Consider a term in the query that is rare in the collection (e.g., *arachnocentric*)
- A document containing this term is very likely to be relevant to the query *arachnocentric*
- → We want a high weight for rare terms like *arachnocentric*.

tf-idf: combine two factors

tf: term frequency. frequency count (usually log-transformed):

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

Idf: inverse document frequency: tf-

Total # of docs in collection

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Words like "the" or "good" have very low idf

# of docs that have word i

tf-idf value for word t in document d:

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

Term Frequency Inverse Document Frequency was introduced in a 1972 paper by Karen Spärck Jones — *"A statistical interpretation of term specificity and its application in retrieval"* (Cambridge, UK)

Words can be vectors too

	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 comedies, especially Twelfth Night"

What do words mean?

Lexical semantics:
the study of word meaning

How do we represent word meaning to use them as:

- input to machine learning models
- applications to other linguistics problems

Words, Lemmas, Senses, Definitions

lemma **sense** **definition**

pepper, *n.*

Pronunciation: Brit. /ˈpɛpə/ , U.S. /ˈpɛpər/

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare*

Frequency (in current use):

Etymology: A borrowing from Latin. **Etymon:** Latin *piper*.

< classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sa

I. The spice or the plant.

1.

a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

The ground spice from *Piper nigrum* comes in two forms, the more pungent *black pepper*, produced from black peppercorns, and the milder *white pepper*, produced from white peppercorns: see BLACK *adj.* and *n.* Special uses 5a, PEPPERCORN *n.* 1a, and WHITE *adj.* and *n.*[1] Special uses 7b(a).

2.

a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

b. Usu. with distinguishing word: any of numerous plants of other families having hot pungent fruits or leaves which resemble pepper (1a) in taste and in some cases are used as a substitute for it.

c. *U.S.* The California pepper tree, *Schinus molle*. Cf. PEPPER TREE *n.*

3. Any of various forms of capsicum, esp. *Capsicum annuum* var. *annuum*. Originally (chiefly with distinguishing word): any variety of the *C. annuum* Longum group, with elongated fruits having a hot, pungent taste, the source of cayenne, chilli powder, paprika, etc., or of the perennial *C. frutescens*, the source of Tabasco sauce. Now frequently (more fully ***sweet pepper***): any variety of the *C. annuum* Grossum group, with large, bell-shaped or apple-shaped, mild-flavoured fruits, usually ripening to red, orange, or yellow and eaten raw in salads or cooked as a vegetable. Also: the fruit of any of these capsicums.

Sweet peppers are often used in their green immature state (more fully ***green pepper***), but some new varieties remain green when ripe.

word

One word can have many senses

The same meaning can be expressed through different words

There are relations between senses

Relation: Synonymy

Synonyms have the same meaning in some or all contexts.

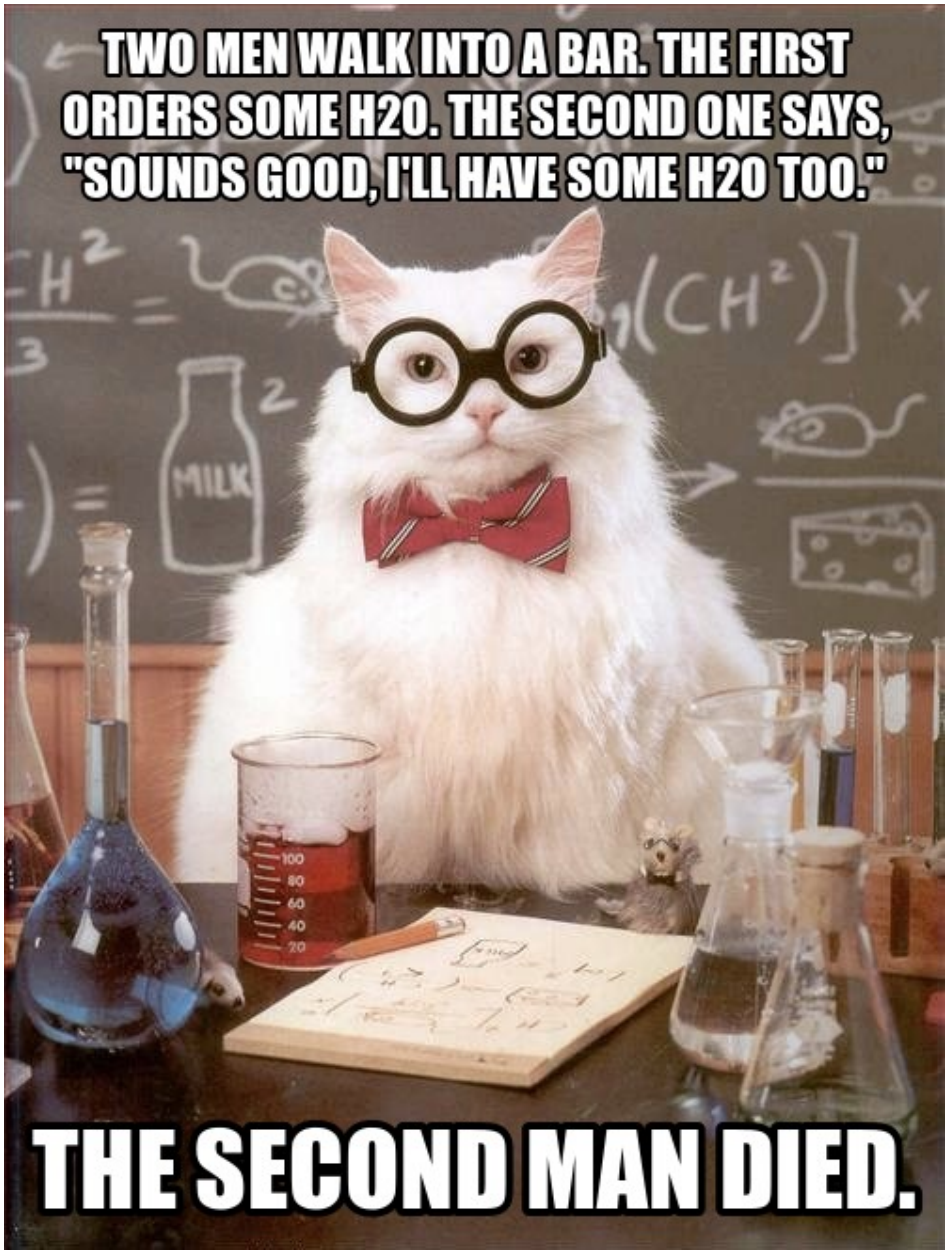
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- Water / H_20

Relation: Synonymy?

Water/H_20

Big/large

Brave/ courageous



Relation: Antonymy

Senses that are opposites with respect to one feature of meaning

Otherwise, they are very similar!

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

`hot/cold  up/down  in/out`

More formally: antonyms can

- define a binary opposition

or be at opposite ends of a scale

- `long/short, fast/slow`
- Be *reversives*:
- `rise/fall, up/down`

Similarity

Words with similar meanings. Not synonyms, but sharing some element of meaning

```
car, bicycle
cow, horse
```

Ask humans how similar 2 words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

More relations

Hypernymy / hyponymy (*tree* / *oak*)
Meronymy / holonymy (*tree* / *branch*)
etc

WordNet : semantic network, curated, free

http://wordnetweb.princeton.edu/perl/webwn
Open Multilingual WordNet (OMW)
https://github.com/dumitrescustefan/RoWordNet

Words can be vectors too

	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 comedies, especially Twelfth Night"

More common: word-word matrix (or "term-context matrix")

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

sugar, a sliced lemon, a tablespoonful of	**apricot**	jam, a pinch each of,
their enjoyment. Cautiously she sampled her first	**pineapple**	and another fruit whose taste she likened
well suited to programming on the digital	**computer**.	In finding the optimal R-stage policy from
for the purpose of gathering data and	**information**	necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

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

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Which pair of words is more similar?

cosine(apricot,information) =

$$\frac{1+0+0}{\sqrt{1+0+0}\ \sqrt{1+36+1}} = \frac{1}{\sqrt{38}} = .16$$

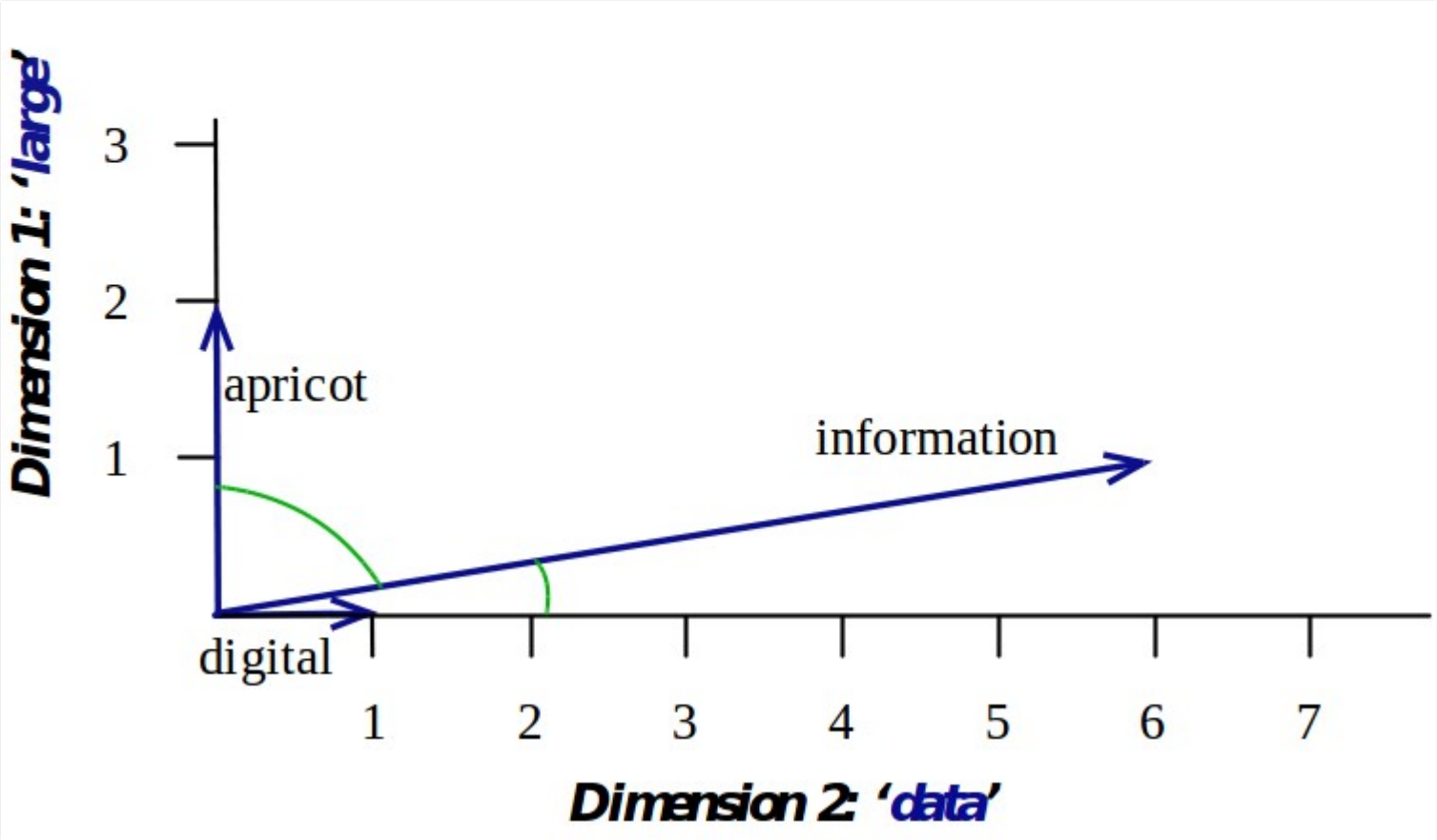
cosine(digital,information) =

$$\frac{0+6+2}{\sqrt{0+1+4}\ \sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

cosine(apricot,digital) =

$$\frac{0+0+0}{\sqrt{1+0+0}\ \sqrt{0+1+4}} = 0$$

Visualizing cosines (well, angles)



From symbolic to distributed word representations

In machine learning vector space terms, this is a vector with one 1 and a lot of zeroes

```
[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]
```

Deep learning people call this a "one-hot" representation

It is a **localist** representation

One-hot-encoding word representations

Its problems, e.g. for web search:

- If a user searches for [Dell notebook battery size], we would like to match documents with "Dell laptop battery capacity"

But

```
size      [0 0 0 0 0 0 0 0 0 1 0 0 0 0]^T
capacity  [0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0
```

Our query and document vectors are **orthogonal**

There is no natural notion of similarity in a set of one hot vectors

One-hot vectors, are

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

Alternative: dense vectors

vectors which are

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

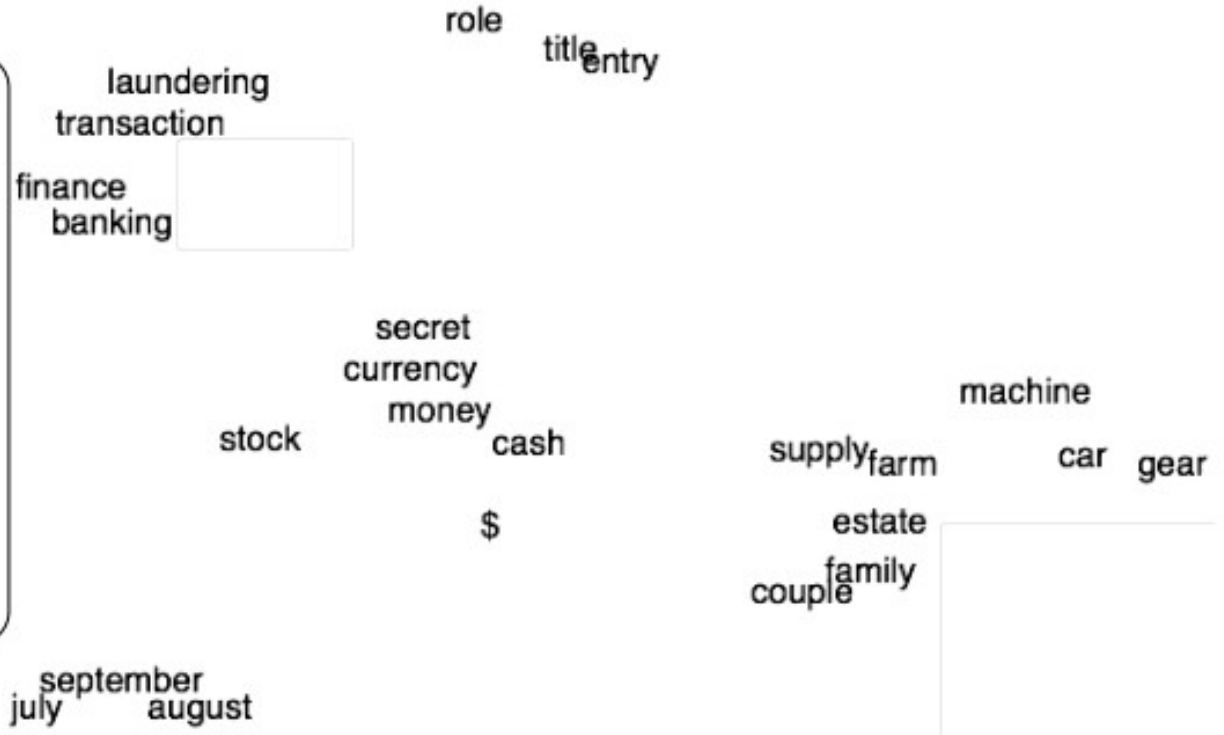
“dense/continuous representations”
“distributed representations“
“word embeddings”

Word meaning as a vector

The result is a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

… those other words also being represented by vectors

$$currency = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



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 storing explicit counts
- They may do better at capturing synonymy:
  - *car* and *automobile* are synonyms; but are distinct dimensions
    - + other semantic relationships (e.g. analogies)

- **In practice, they work better**

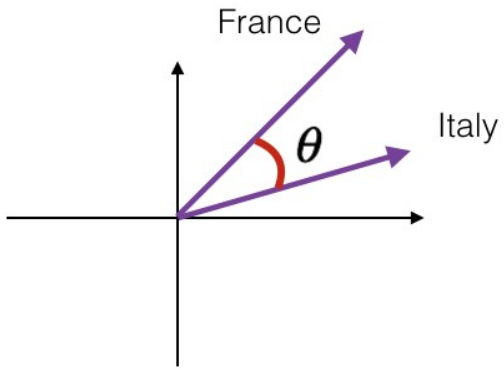
Capturing similarity

Query: *fast streams*

Document: *Dambovita is a very rapid river*

Orthogonal vectors: no common words (low similarity in tf-idf vector space) - but high semantic similarity!

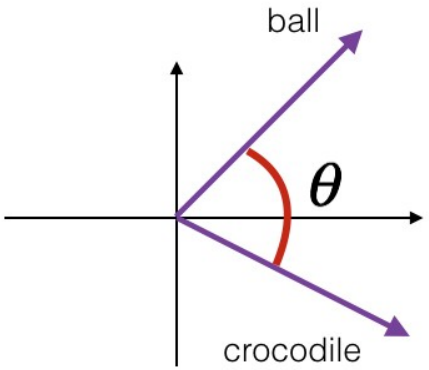
Word similarity: cosine distance



France and Italy are quite similar

θ is close to 0°

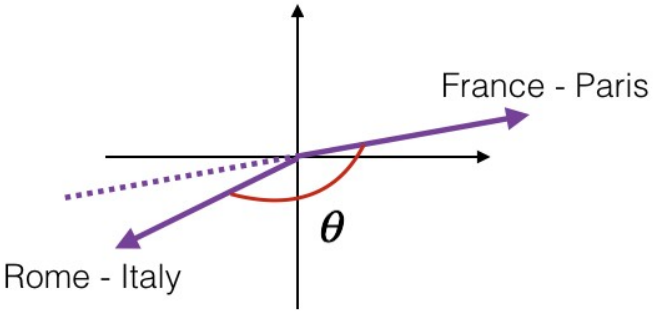
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$



the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

$\cos(\theta) \approx -1$

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

A solution via distributional similarity-based representations

Idea: representing a word by means of its neighbors / context

"You shall know a word by the company i

(J. R. Firth, 1957)

Philosophy: Ludwig Wittgenstein

"The meaning of a word is defined by the way it is used"

What does ongchoi mean?

Suppose you see these sentences:

- Ong choi is delicious **sautéed with garlic**.
- Ong choi is superb **over rice**
- Ong choi **leaves** with salty sauces

And you've also seen these:

- ...spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:

- Ongchoi is a leafy green like spinach, chard, or collard greens

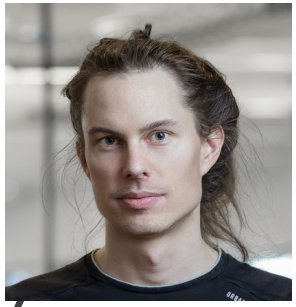
Ong choi: *Ipomoea aquatica* "*Water Spinach*"

Yamaguchi, Wikimedia Commons, public domain

Brilliant insight: Use running text as implicitly supervised training data!

- A word *s* near *apricot*
  - Acts as gold ‘correct answer’ to the question
  - “Is word *w* likely to show up near *apricot*?”
- No need for hand-labeled supervision
- The idea comes from **neural language modeling**
  - Bengio et al. (2003)
  - Collobert et al. (2011)

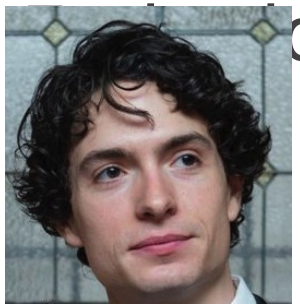
Dense embeddings you can download!



Word2vec (Mikolov et al., 2013, https://code.google.com/archive/p/word2vec/

Fasttext http://www.fasttext.cc/ (sub-word information)

Glove (Pennington, Socher, Manning, 2014)
http://nlp.stanford.edu/projects/glove/



erything!

Dense embeddings you can download!

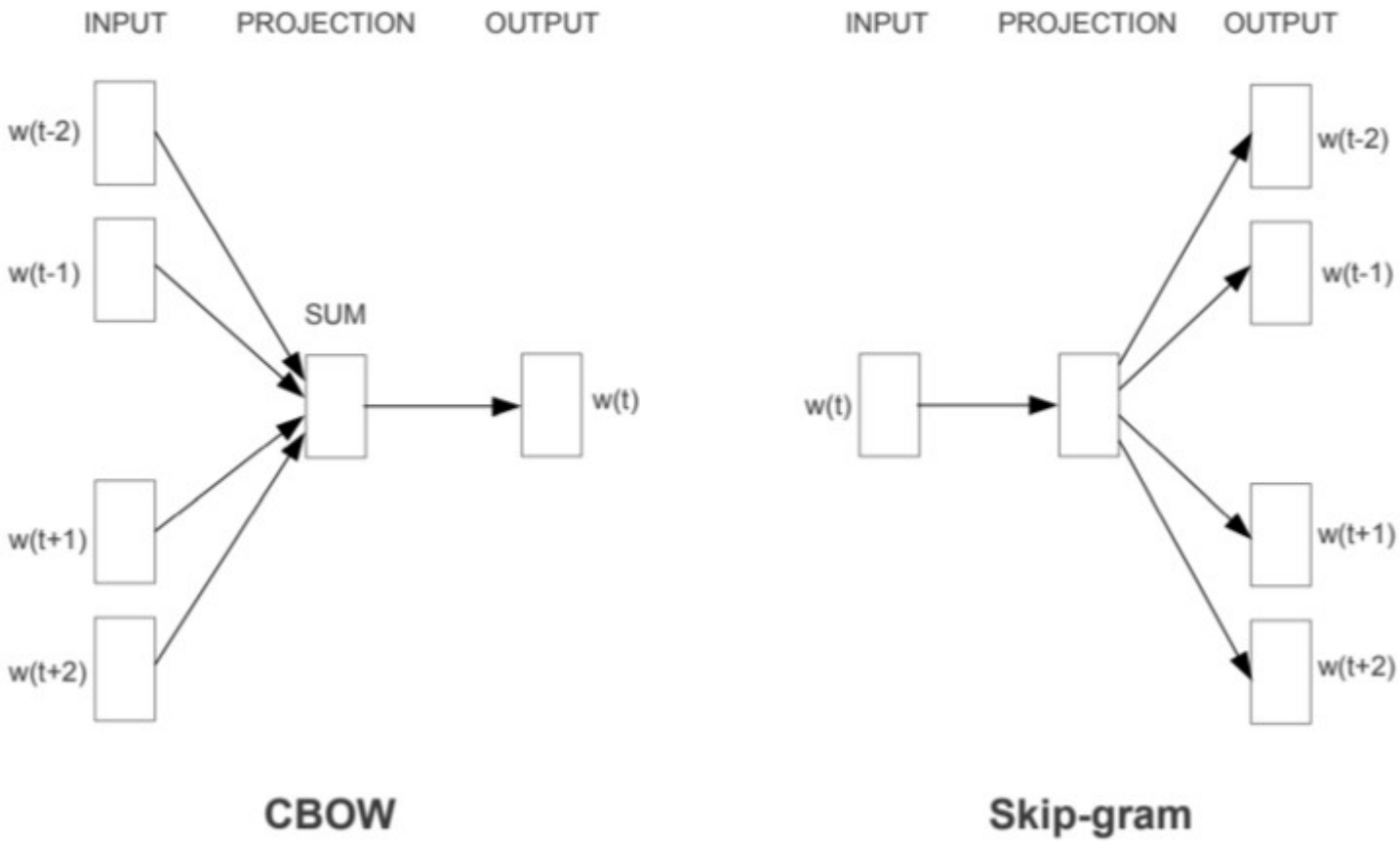
Romanian (& many more languages):

https://fasttext.cc/docs/en/pretrained-vectors.html

Word2vec

Popular embedding method
Very fast to train
Code available on the web
Idea: **predict** rather than **count**

Word2vec variants



Skip-gram with negative sampling (SNGS)

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 weights as the

Context windows

Source Text	Training Samples generated from source text			
I will have orange juice and eggs for breakfast	(will, I)	(will, have)	(will, orange)	
I will have orange juice and eggs for breakfast	(have, I)	(have, will)	(have, orange)	(have, juice)
I will have orange juice and eggs for breakfast	(orange, will)	(orange, have)	(orange, juice)	(orange, and)
I will have orange juice and eggs for breakfast	(juice, have)	(juice, orange)	(juice, and)	(juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange)	(and, juice)	(and, eggs)	(and, for)
I will have orange juice and eggs for breakfast	(eggs, juice)	(eggs, and)	(eggs, for)	(eggs, breakfast)
I will have orange juice and eggs for breakfast	(for, and)	(for, eggs)	(for, breakfast)	

Summary: How to learn word2vec (skip-gram) embeddings

Start with V random 300-dimensional vectors as initial embeddings
Use logistic regression, the second most basic classifier used in machine learning after naïve bayes

- Take a corpus and take pairs of words that co-occur as positive examples
- Take pairs of words that don't co-occur as negative examples
- Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
- Throw away the classifier code and keep the

Evaluating embeddings

Intrinsic:
Compare to human scores on word similarity-type tasks:

- WordSim-353 (Finkelstein et al., 2002)
- SimLex-999 (Hill et al., 2015)
- Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)
- TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

Properties of embeddings

Similarity depends on window size C

C = ±2 The nearest words to *Hogwarts:*

- *Sunnydale*
- *Evernight*

C = ±5 The nearest words to *Hogwarts:*

- *Dumbledore*
- *Malfoy*
- *halfblood*

Properties of embeddings

Results also depend on:

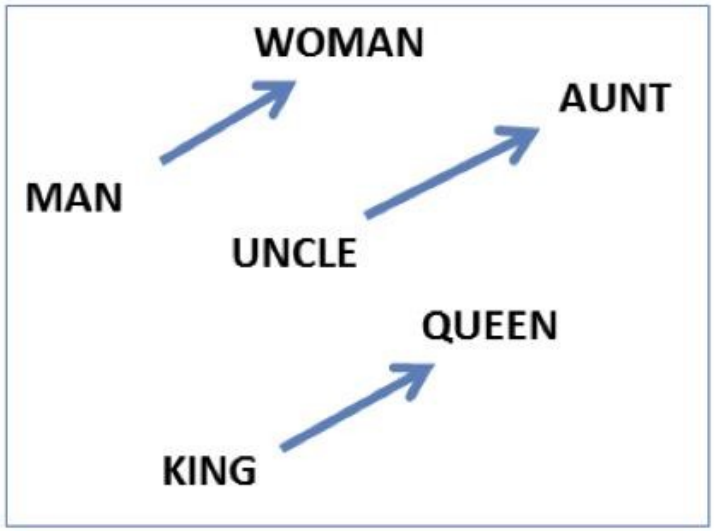
- the corpus used for training the embeddings - should ideally be similar to your data, with enough data you can also train your own
- their dimensionality (50/100/300/...)

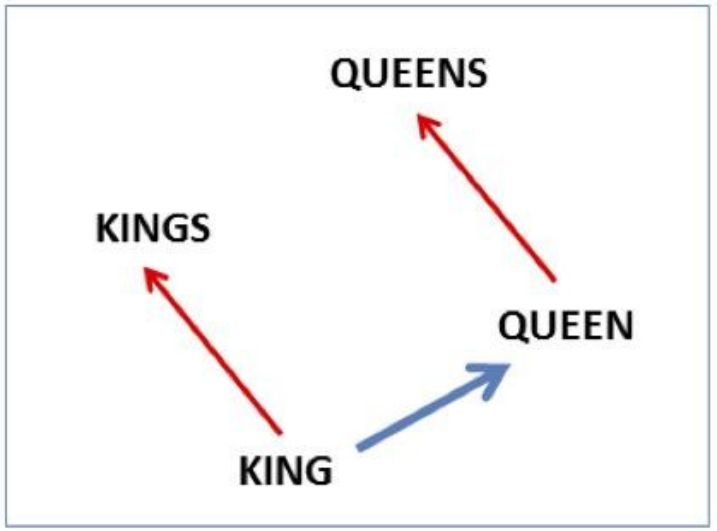
Visualizing embeddings

https://projector.tensorflow.org/

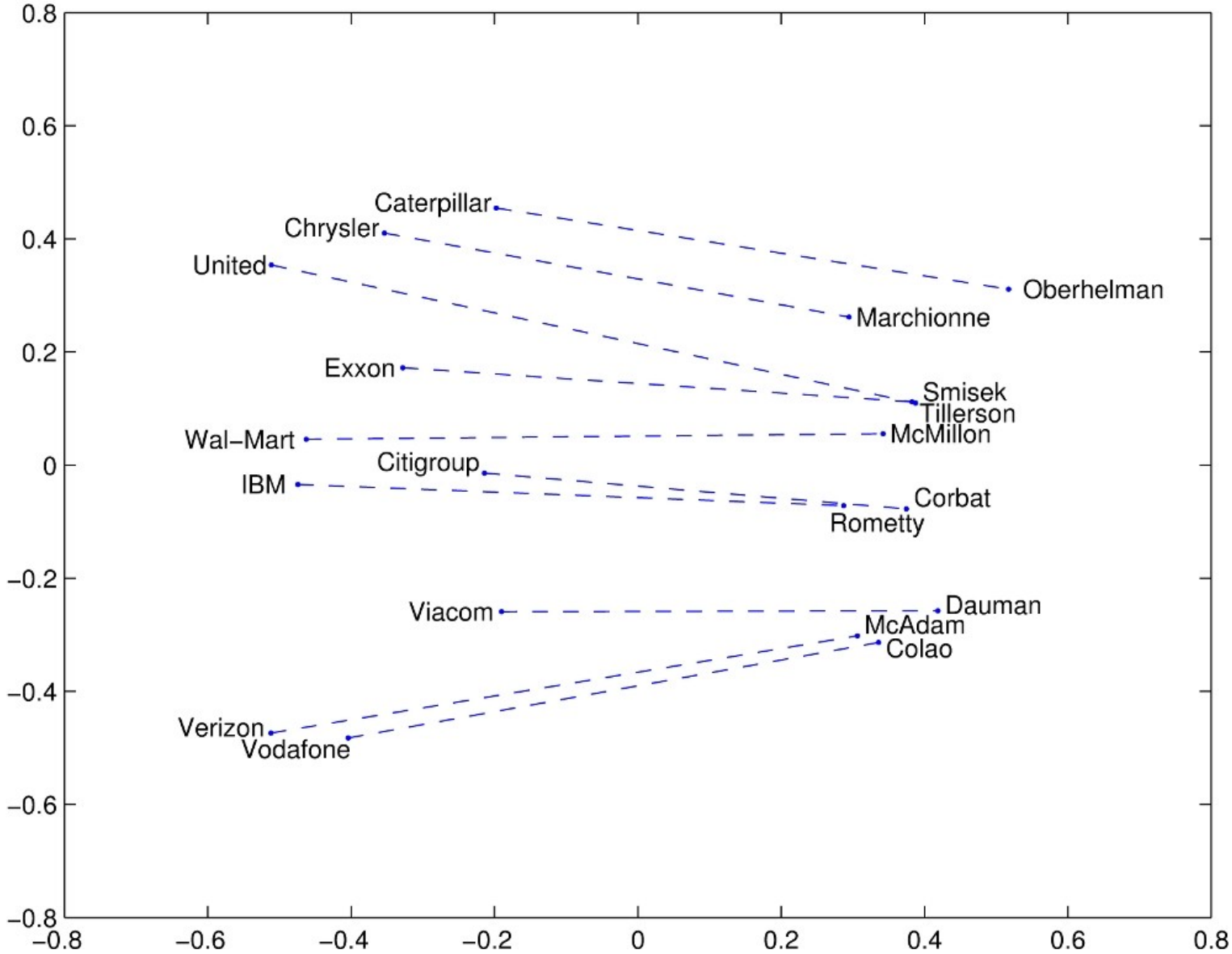
Analogy: Embeddings capture relational meaning!

vector(‘*king*’) - vector(‘*man*’) + vector(‘*woman*’) ≈ vector(‘queen’)
vector(‘*Paris*’) - vector(‘*France*’) + vector(‘*Italy*’) ≈ vector(‘Rome’)

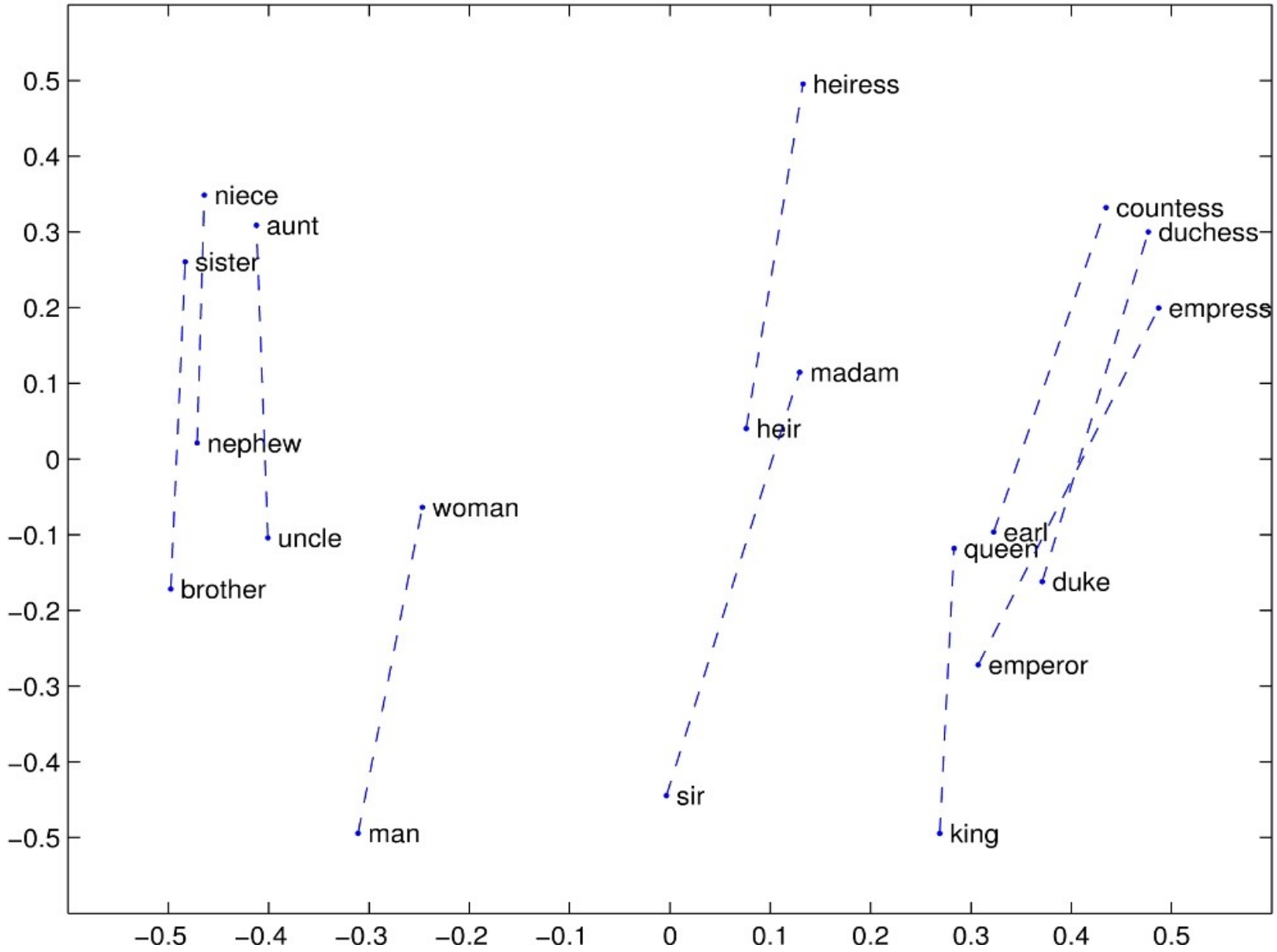




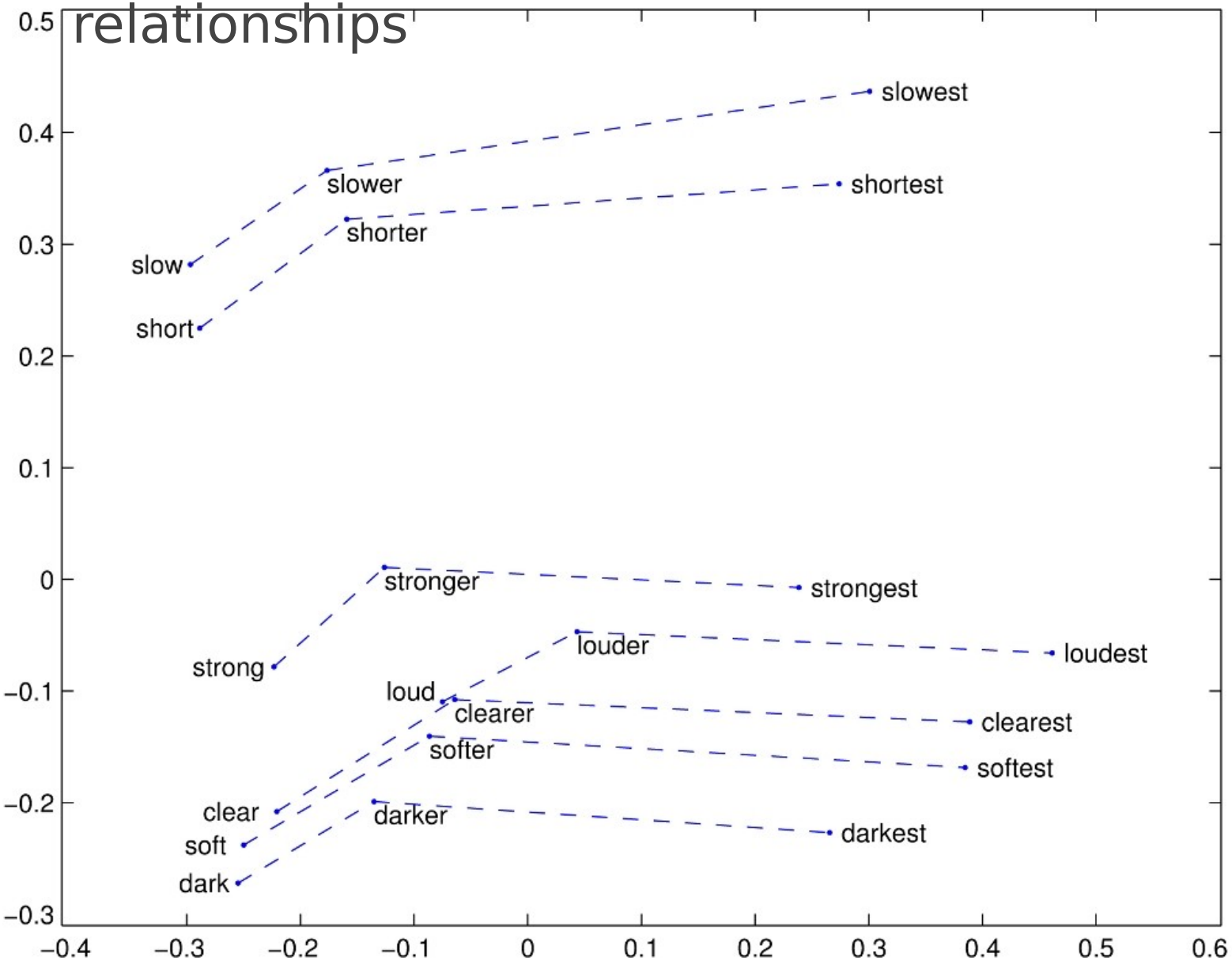
GloVe visualizations: company-CEO



GloVe visualizations: gender pairs



GloVe visualizations: morphological relationships



Embeddings applications

- Word Similarity
- Machine Translation
- Part-of-Speech and Named Entity Recognition
- Sentiment Analysis
- Semantic Analysis of Documents
  - Build word distributions for various topics, etc.
- Text classification, other ML models... (input to classifiers - use pretrained embeddings out-of-the-box / fine tune them)
- Any NLP application with a semantic component...?
- Linguistics applications: ⟹

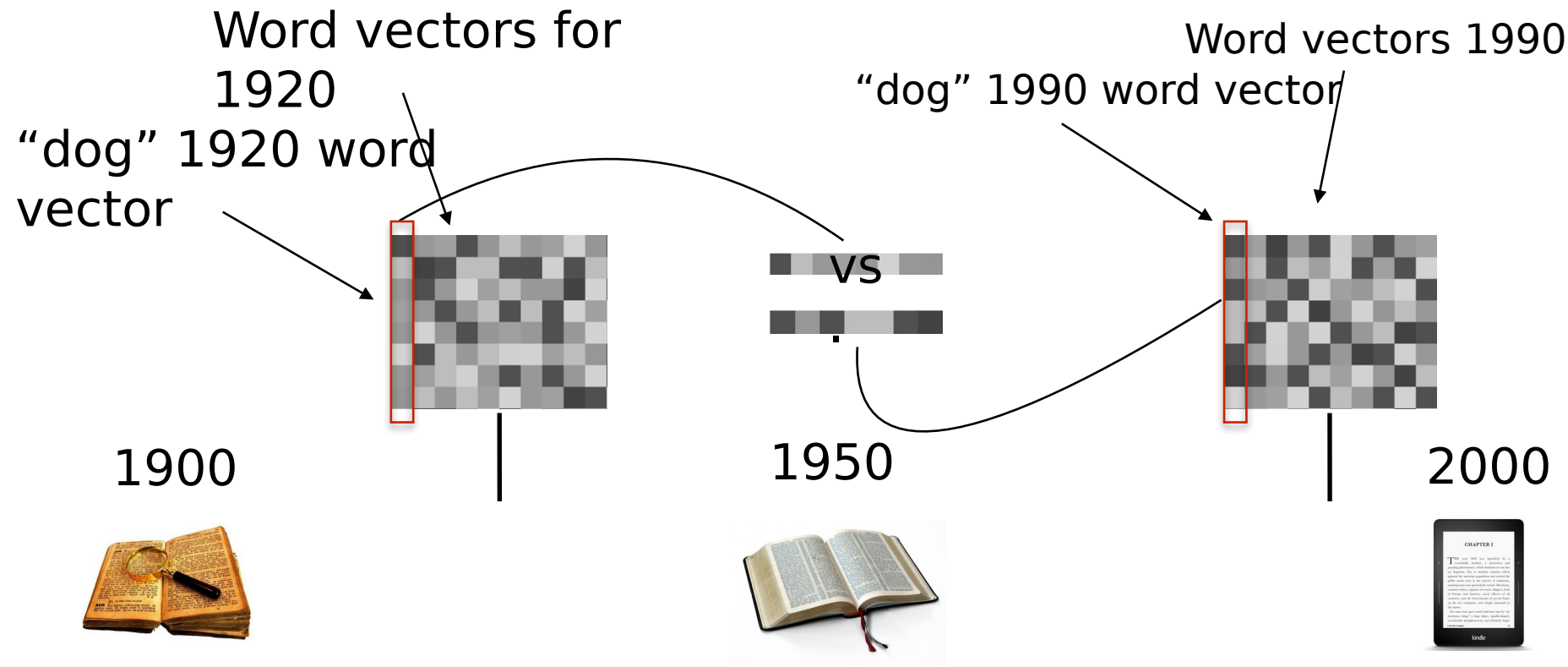
Embeddings can help study word history!

Train embeddings on old books to study changes in word meaning!

Idea: train embeddings on two different corpora; compare

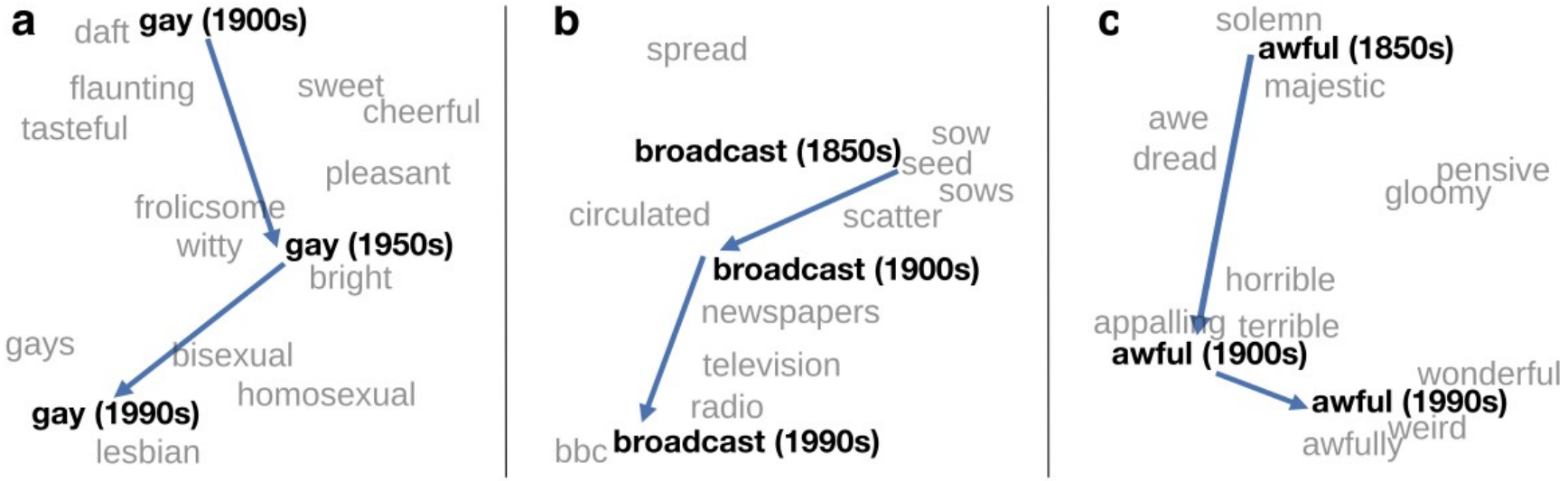
Other applications: compare word connotations in different domains, dialects

embeddings for studying language change!



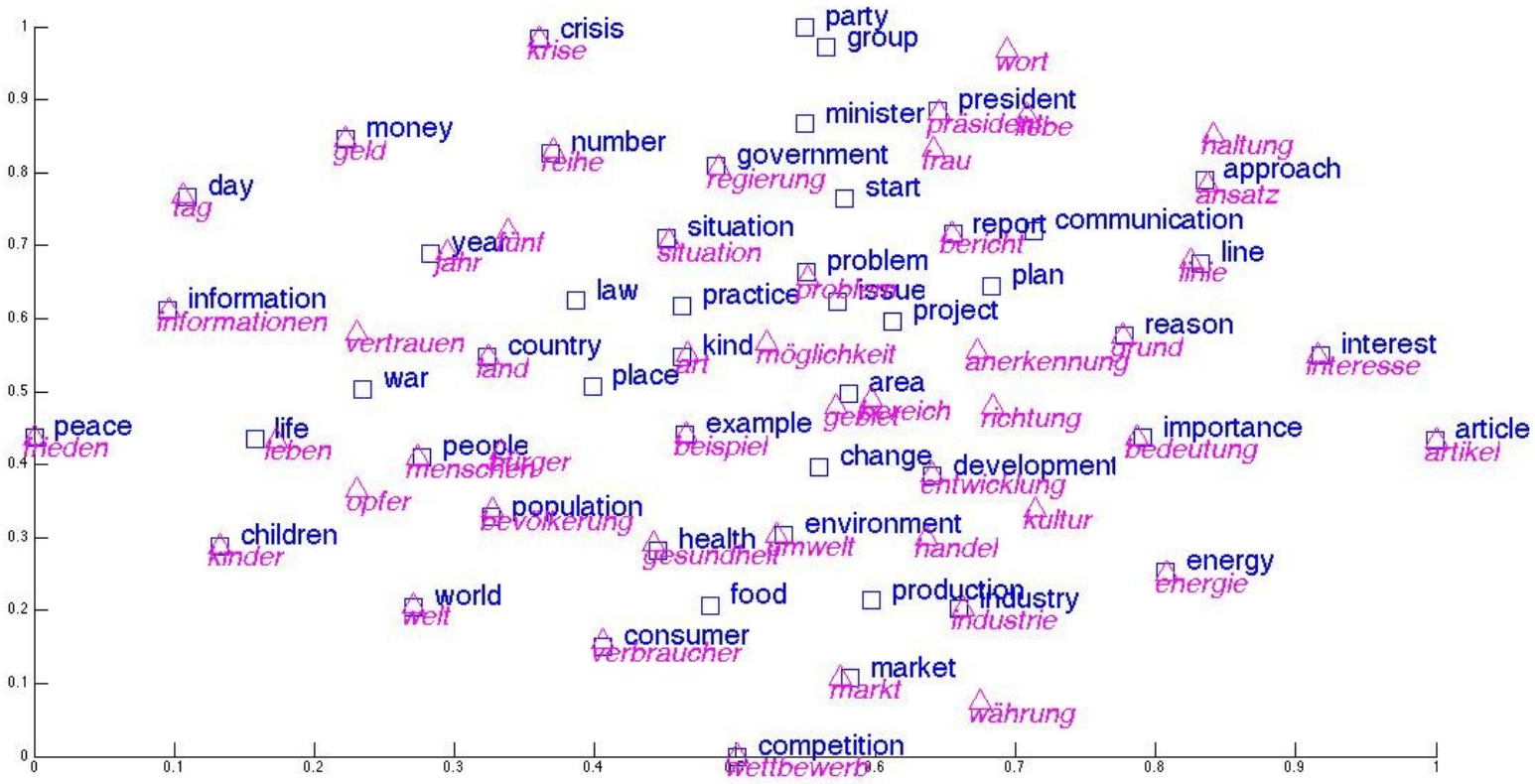
Visualizing changes

Project 300 dimensions down into 2



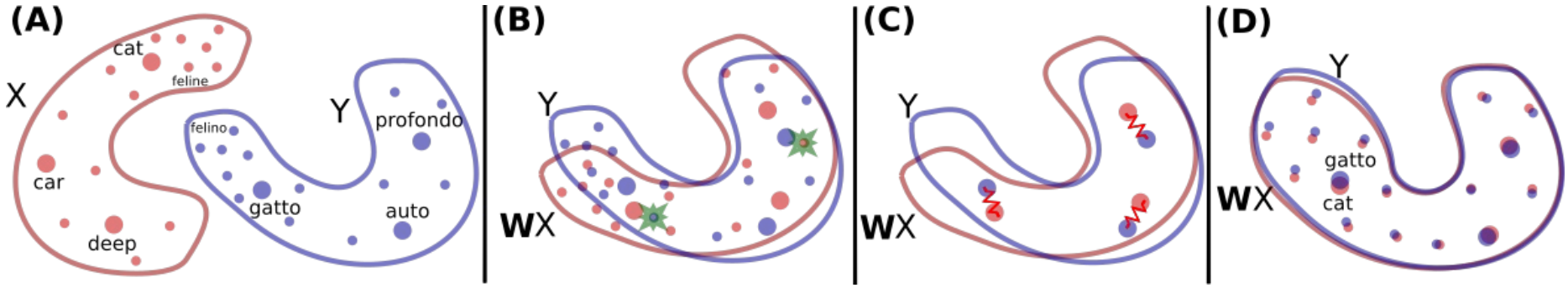
~30 million books, 1850-1990, Google Books data

Multilingual embeddings



Multilingual embeddings

Can be learned in an unsupervised way, through alignment of monolingual spaces



https://github.com/facebookresearch/MUSE

Multilingual embeddings

We can use multilingual embeddings to detect false friends!
https://github.com/ananana/false_friends_resource

embarazada ... embarrassed ??

History of biased framings of women

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

Embeddings for competence adjectives are biased toward men

- *Smart, wise, brilliant, intelligent, resourceful, thoughtful, logical, etc.*

This bias is slowly decreasing

Embeddings reflect cultural bias

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *Advances in Neural Information Processing Systems*, pp. 4349-4357. 2016.

Ask “Paris : France :: Tokyo : x”

- x = Japan

Ask “father : doctor :: mother : x”

- x = nurse

Ask “man : computer programmer :: woman : x”

- x = homemaker

Embeddings as a window onto history

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

Use the Hamilton historical embeddings
The cosine similarity of embeddings for decade X for occupations (like teacher) to male vs female names

- Is correlated with the actual percentage of women teachers in decade X

Embeddings reflect ethnic stereotypes over time

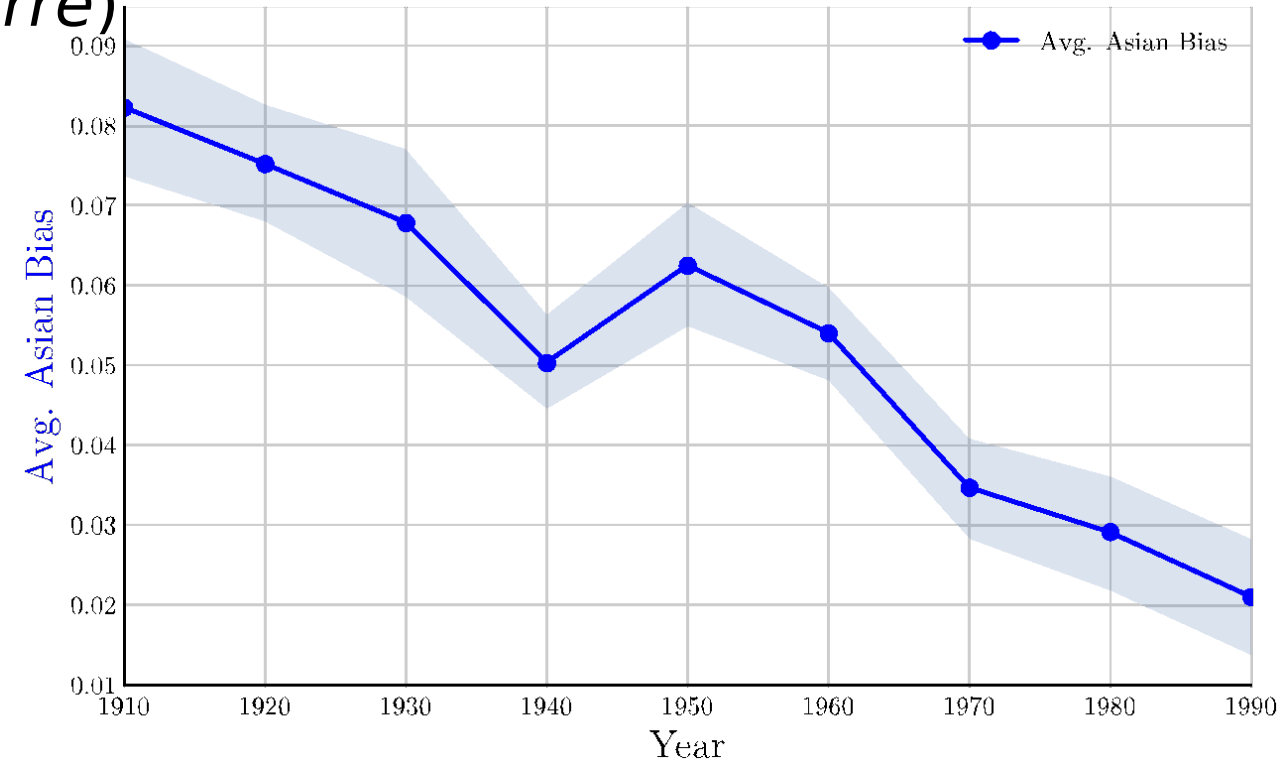
Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

- Princeton trilogy experiments
- Attitudes toward ethnic groups (1933, 1951, 1969) scores for adjectives
  - *industrious, superstitious, nationalistic*, etc
- Cosine of Chinese name embeddings with those adjective embeddings correlates with human ratings.

Change in linguistic framing 1910-1990

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

Change in association of Chinese names with adjectives framed as "othering" (*barbaric*, *monstrous*, *bizarre*)



Changes in framing: adjectives associated with Chinese

Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, *115*(16), E3635–E3644

1910	1950	1990
Irresponsible	Disorganized	Inhibited
Envious	Outrageous	Passive
Barbaric	Pompous	Dissolute
Aggressive	Unstable	Haughty
Transparent	Effeminate	Complacent
Monstrous	Unprincipled	Forceful
Hateful	Venomous	Fixed
Cruel	Disobedient	Active
Greedy	Predatory	Sensitive
Bizarre	Boisterous	Hearty

Context is key

- Language is complex, and **context** can completely change the meaning of a word in a sentence.
- Example:
  - I let the kids outside to **play**.
  - He had never acted in a more famous **play** before.
  - It wasn't a **play** the coach would approve of.
- Need a model which captures the different nuances of the meaning of words given the surrounding text.

Beyond word2vec:
Contextual embeddings

ELMo: Embeddings from Language Models Representations

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations."

BERT

J.Devlin, M. Chang, K. Lee and K. Toutanova,
BERT: Pre-training of Deep Bidirectional Transformers for Language nding
(2018)

ChatGPT...

Difference to other methods

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {. . . }	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {. . . }	{. . . } they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

§ Nearest neighbors words to “play” using GloVe and the nearest neighbor sentences to “play” using ELMo.

Practical examples in Python: word embeddings

https://colab.research.google.com/drive/1u0fJJpLMHjONG7umdBSIFfwm29nvj_nE?usp=sharing

Thank you!

Good luck with the hackathon and come join us on the
😇 NLP side !
https://nlp.unibuc.ro/master_en.html