Language Technologies Institute

Carnegie Mellon University

Algorithms for NLP

CS 11-711 · Fall 2020

Lecture 4: Vector semantics and word embeddings

Emma Strubell

Announcements

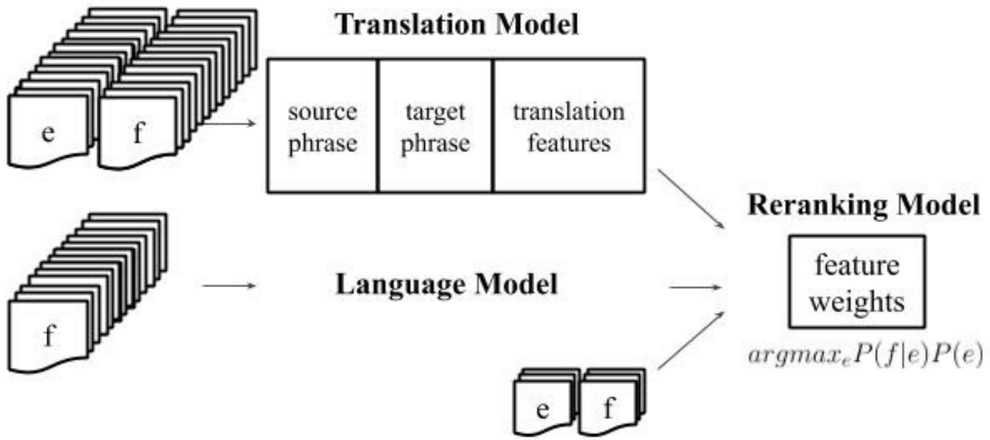
- Don’t forget we will hold recitations Friday to discuss P1.
- Working on getting lectures posted on YouTube (CMU-only) w/ closed captions.

Recap

Nonlinear classification & evaluating classifiers

Engineered features

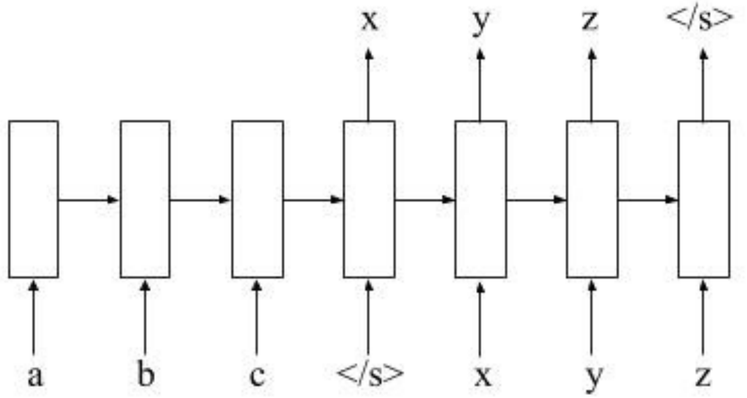
linear classification



~mid 2010s

Learned features

nonlinear classification



Recap

Another choice of $\mathcal{R}$: word embeddings

- Text is naturally viewed as a sequence of tokens w_1, w_2, …, w_T
- Context is lost when this sequence is converted to a bag-of-words.
- Instead, a **lookup layer** can compute **embeddings** (real-valued vectors) for each type, resulting in a matrix

$\boldsymbol{w}$ =

-1.36	0.09	0.71	-1.57	-0.72	-1.36	0.45	0.71	0.23
-0.23	-1.69	1.43	-0.42	-0.08	-0.23	-2.55	1.43	0.21
0.84	-0.16	0.11	0.54	1.56	0.84	-0.39	0.11	1.11
…	…	…	…	…	…	…	…	…
-0.067	0.71	-5.6	0.26	0.07	-0.067	-0.77	-5.6	0.55
the	drinks	were	strong	but	the	tacos	were	bland

How to represent a word?

How to represent a word?

- Linguistic perspective: **lexical semantics**.

How to represent a word?

- Linguistic perspective: **lexical semantics**.

pepper, *n.*

View as: Outline | Full entry

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

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare*), OE–15 **piper**, ME **papeer**, ME **paupere**, ME **peopur**, ME **pepir**, ME **pepire**, ME **pepre**, ME **pepur**, ME **pepyr**, ME **pyper**, ME–15 **papur**, ME–17 **pep** error), 18 **pipper** (*Irish English*); *English regional* (*East Anglian*) 18– **pupper**; *Scottish* pre-17 **pepar**, pre-17 **peper**, pre-1 17 **pepyre**, pre-17 **piper**, pre-17 17– **pepper**. *N.E.D.* (1905) also records a form ME **pepyre**. (Show Less)

Frequency (in current use):

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

Etymology: < classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sanskrit *pippalī* long p French, Middle French, French *poivre* (*c*1140 in Old French as *peivere* : see POIVRADE *n.*), Old Occitan, Occitan *pebre* (*a*1126),

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.

b. With distinguishing word: any of certain other pungent spices derived from plants of other families, *esp.* ones used as seasonings.

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.

How to represent a word?

- Linguistic perspective: **lexical semantics**.

pepper, *n.* ← **lemma**

View as: Outline | Full entry

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

Forms: OE **peopor** (*rare*), OE **pipcer** (transmission error), OE **pipor**, OE **pipur** (*rare*), OE–15 **piper**, ME **papeer**, ME **paupere**, ME **peopur**, ME **pepir**, ME **pepire**, ME **pepre**, ME **pepur**, ME **pepyr**, ME **pyper**, ME–15 **papur**, ME–17 **pep** error), 18 **pipper** (*Irish English*); *English regional* (*East Anglian*) 18– **pupper**; *Scottish* pre-17 **pepar**, pre-17 **peper**, pre-1 17 **pepyre**, pre-17 **piper**, pre-17 17– **pepper**. *N.E.D.* (1905) also records a form ME **pepyre**. (Show Less)

Frequency (in current use): ●●●●●●○○

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

Etymology: < classical Latin *piper*, a loanword < Indo-Aryan (as is ancient Greek *πέπερι*); compare Sanskrit *pippalī* long p French, Middle French, French *poivre* (*c*1140 in Old French as *peivere* : see POIVRADE *n.*), Old Occitan, Occitan *pebre* (*a*1126),

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.

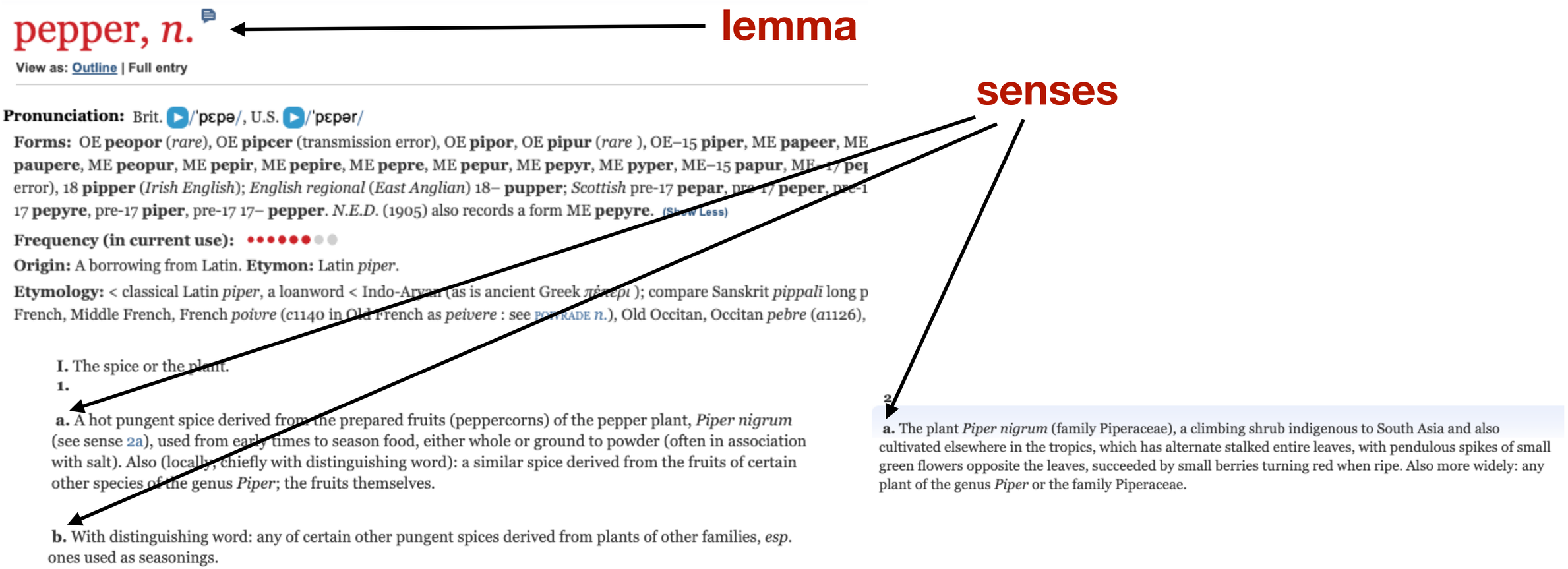
b. With distinguishing word: any of certain other pungent spices derived from plants of other families, *esp.* ones used as seasonings.

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.

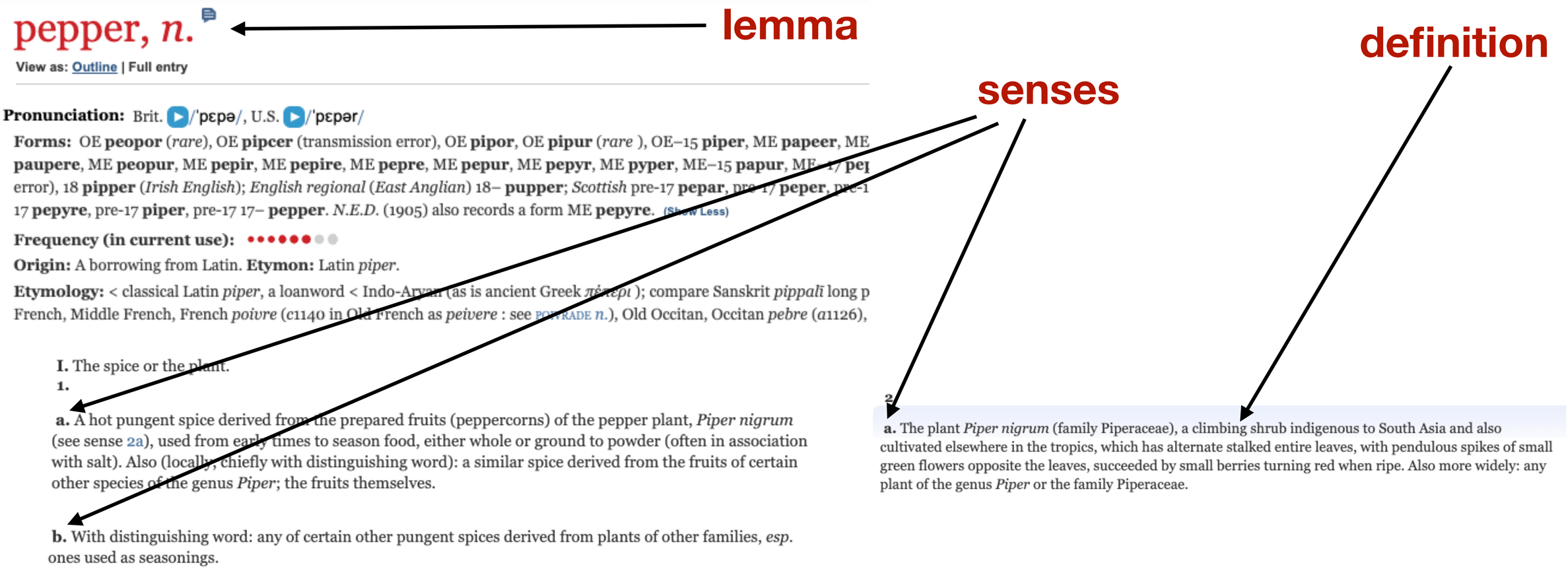
How to represent a word?

- Linguistic perspective: **lexical semantics**.



How to represent a word?

- Linguistic perspective: **lexical semantics**.



How to represent a word?

Lexical semantics

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

    1. spice from the pepper plant

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

    1. spice from the pepper plant
    2. the pepper plant itself

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

    1. spice from the pepper plant
    2. the pepper plant itself
    3. another similar plant (Jamaican pepper)

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

    1. spice from the pepper plant
    2. the pepper plant itself
    3. another similar plant (Jamaican pepper)
    4. another plant with peppercorns (California pepper)

A sense or “concept” is the meaning component of a word.

How to represent a word?

Lexical semantics

- Senses of the lemma *pepper*:

    1. spice from the pepper plant
    2. the pepper plant itself
    3. another similar plant (Jamaican pepper)
    4. another plant with peppercorns (California pepper)
    5. capsicum (i.e. chili, paprila, bell pepper, etc.)

A sense or “concept” is the meaning component of a word.

Relations between words

Synonymy

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - automobile / car

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - automobile / car
  - water / H_2O

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - automobile / car
  - water / H_2O
- Note that there are probably **no** examples of perfect synonymy

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - automobile / car
  - water / H_2O
- Note that there are probably **no** examples of perfect synonymy
  - Even if many aspects of meaning are identical

Relations between words

Synonymy

- **Synonyms** have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - automobile / car
  - water / H_2O
- Note that there are probably **no** examples of perfect synonymy
  - Even if many aspects of meaning are identical
  - Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.

Relations between words

Antonymy

Relations between words

Antonymy

- **Antonyms**: senses that are opposite with respect to one feature of meaning

Relations between words

Antonymy

- **Antonyms**: senses that are opposite with respect to one feature of meaning
- Otherwise, they are very similar!

Relations between words

Antonymy

- **Antonyms**: senses that are opposite with respect to one feature of meaning
- Otherwise, they are very similar!
  - dark / light; short / long; fast / slow; rise / fall

Relations between words

Antonymy

- **Antonyms**: senses that are opposite 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

Relations between words

Antonymy

- **Antonyms**: senses that are opposite 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:

Relations between words

Antonymy

- **Antonyms**: senses that are opposite 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

Relations between words

Antonymy

- **Antonyms**: senses that are opposite 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

Relations between words

Similarity

Relations between words

Similarity

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

Relations between words

Similarity

- Words with similar meanings. Not synonyms, but sharing some element of meaning.
  - car / bicycle; cow / horse; running / jogging

Relations between words

Similarity

- Words with similar meanings. Not synonyms, but sharing some element of meaning.
    - car / bicycle; cow / horse; running / jogging
- What happens when we ask humans how similar words are?

Relations between words

Similarity

- Words with similar meanings. Not synonyms, but sharing some element of meaning.
  - car / bicycle; cow / horse; running / jogging
- What happens when we ask humans how similar words are?

word$_1$	**word$_2$**	**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].

Relations between words

Semantic field

Relations between words

Semantic field

- Words that:

Relations between words

Semantic field

- Words that:
  - cover a particular semantic domain

Relations between words

Semantic field

- Words that:
  - cover a particular semantic domain
  - bear structured relations with each other

Relations between words

Semantic field

- Words that:
  - cover a particular semantic domain
  - bear structured relations with each other

hospitals

surgeon, scalpel, nurse, anesthetic, hospital

Relations between words

Semantic field

- Words that:
  - cover a particular semantic domain
  - bear structured relations with each other

hospitals

surgeon, scalpel, nurse, anesthetic, hospital

restaurants

waiter, menu, plate, food, chef, drink

Relations between words

Semantic field

- Words that:
  - cover a particular semantic domain
  - bear structured relations with each other

hospitals

surgeon, scalpel, nurse, anesthetic, hospital

restaurants

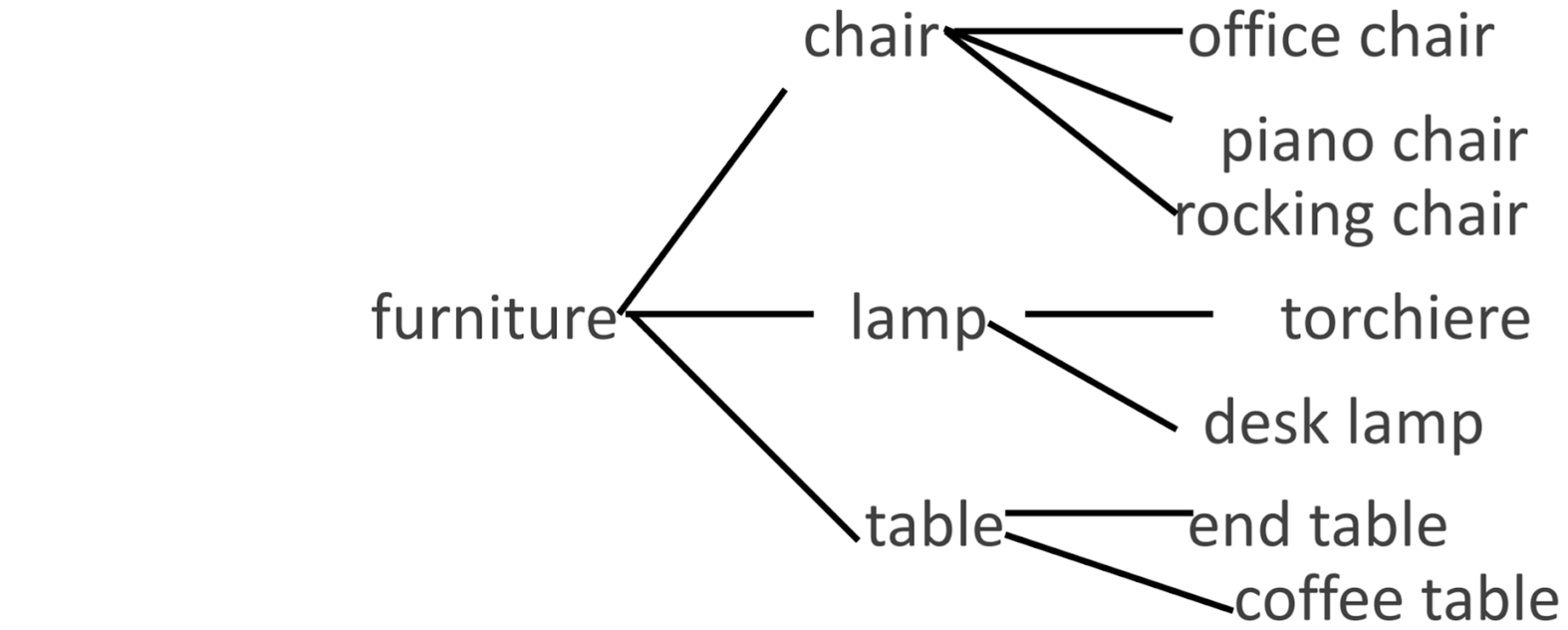
waiter, menu, plate, food, chef, drink

houses

door, roof, kitchen, family, bed

Relations between words

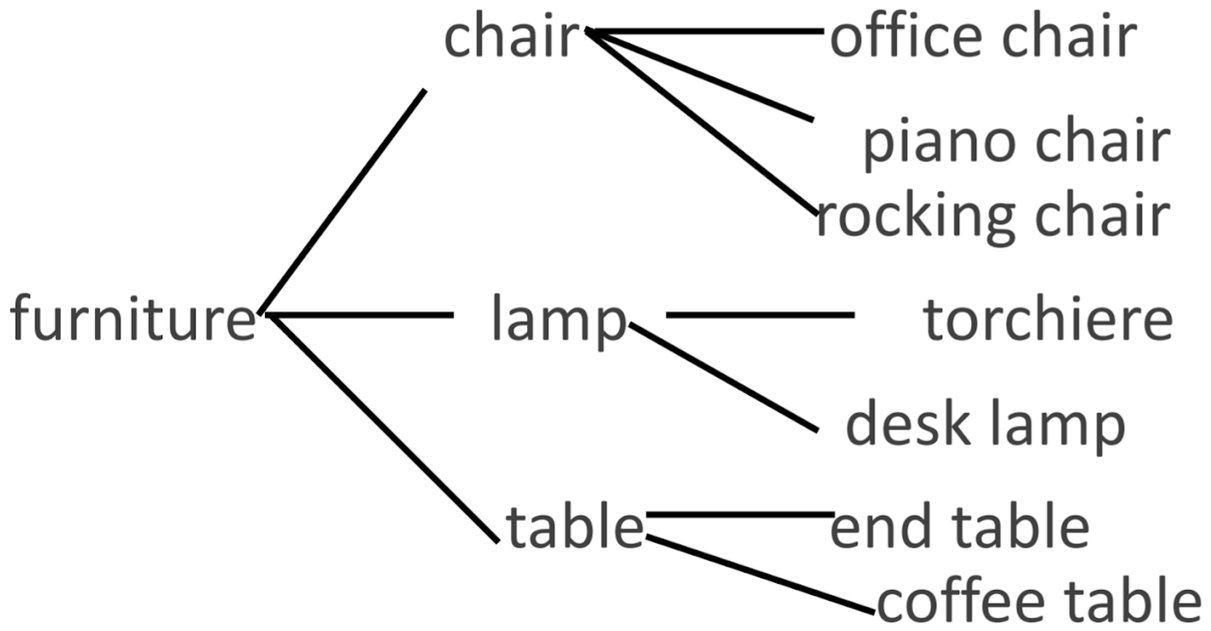
Hypernym / hyponym (superordinate / subordinate)



Relations between words

Hypernym / hyponym (superordinate / subordinate)

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other

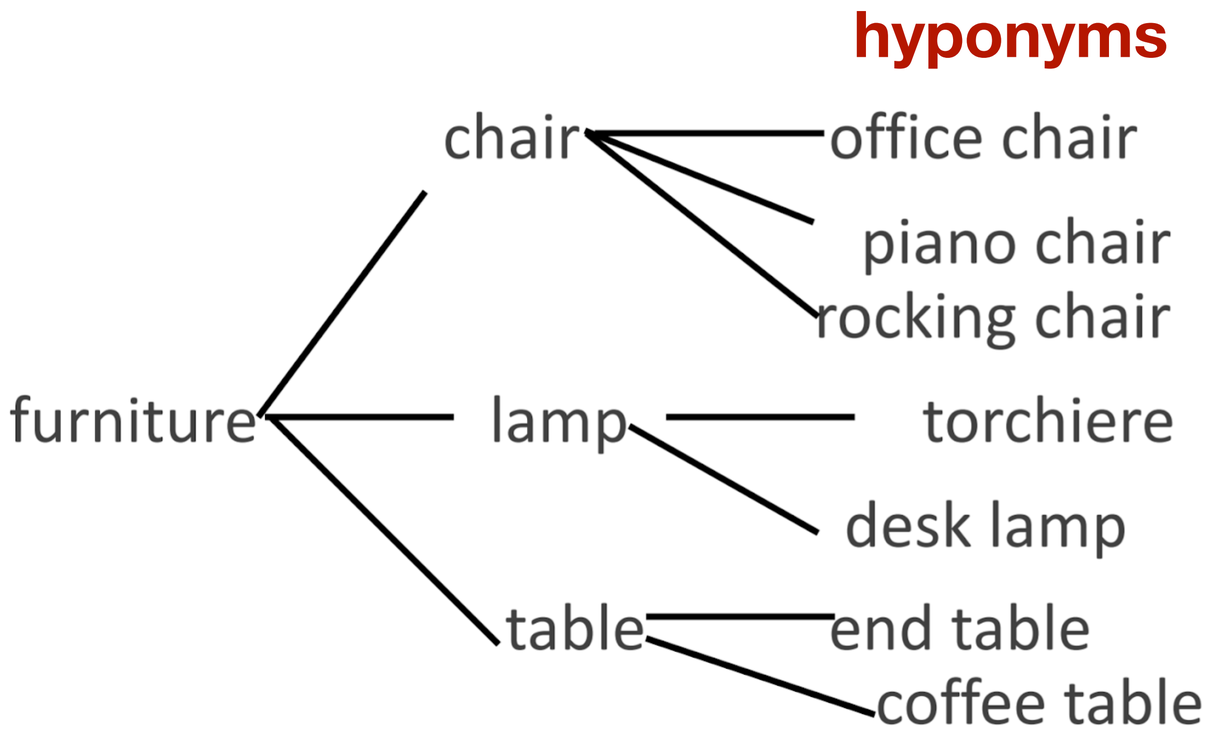


Relations between words

Hypernym / hyponym (superordinate / subordinate)

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other

Taxonomy:

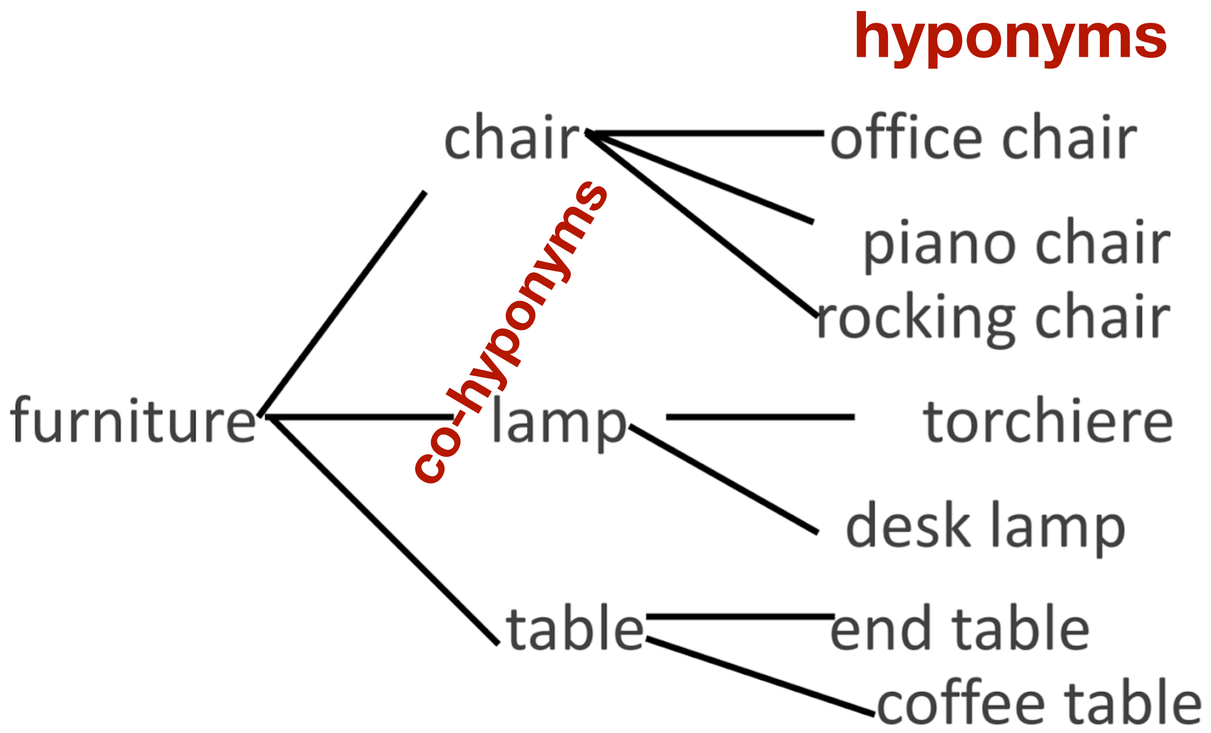


Relations between words

Hypernym / hyponym (superordinate / subordinate)

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other

Taxonomy:

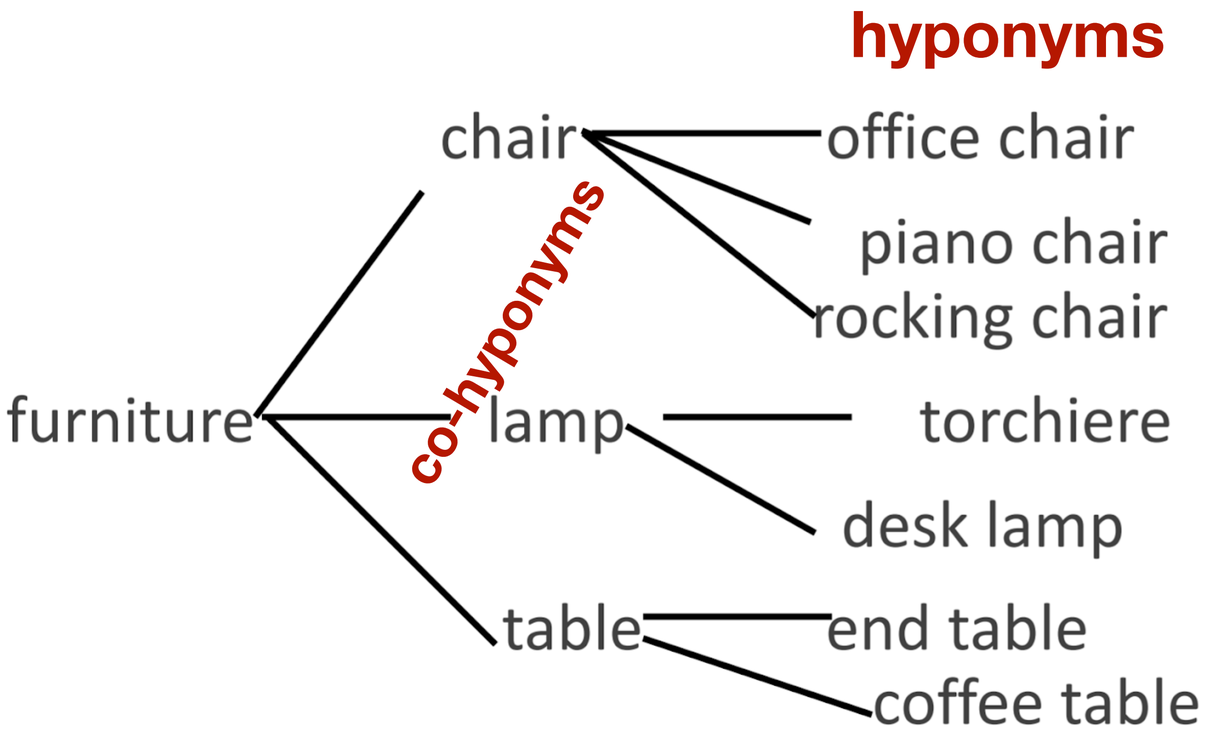


Relations between words

Hypernym / hyponym (superordinate / subordinate)

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other
- Conversely, **hypernymy**.

Taxonomy:

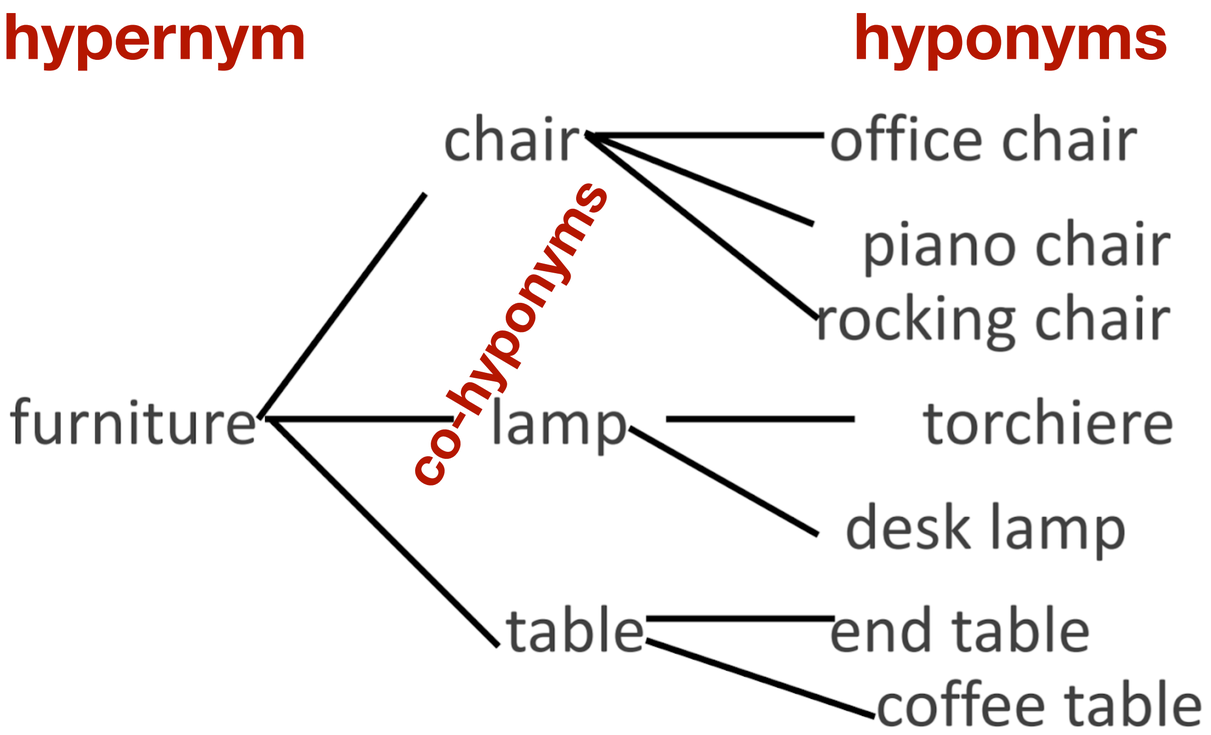


Relations between words

Hypernym / hyponym (superordinate / subordinate)

- One sense is a **hyponym** of another if the first sense is more specific, denoting a subclass of the other
- Conversely, **hypernymy**.

Taxonomy:



How to represent the meaning of a word?

Lexical semantics

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles, e.g. *Sam bought the book from Ling.*
  Ling sold the book to Sam.

How to represent the meaning of a word?

Lexical semantics

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

	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

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

pleasantness

	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

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

pleasantness intensity

	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

How to represent the meaning of a word?

Lexical semantics

- Dictionary definition
- Lemma and word forms
- Senses
- Relationships between words or senses
- Taxonomic relationships
- Word similarity, word relatedness
- Semantic frames and roles
- Connotation and sentiment

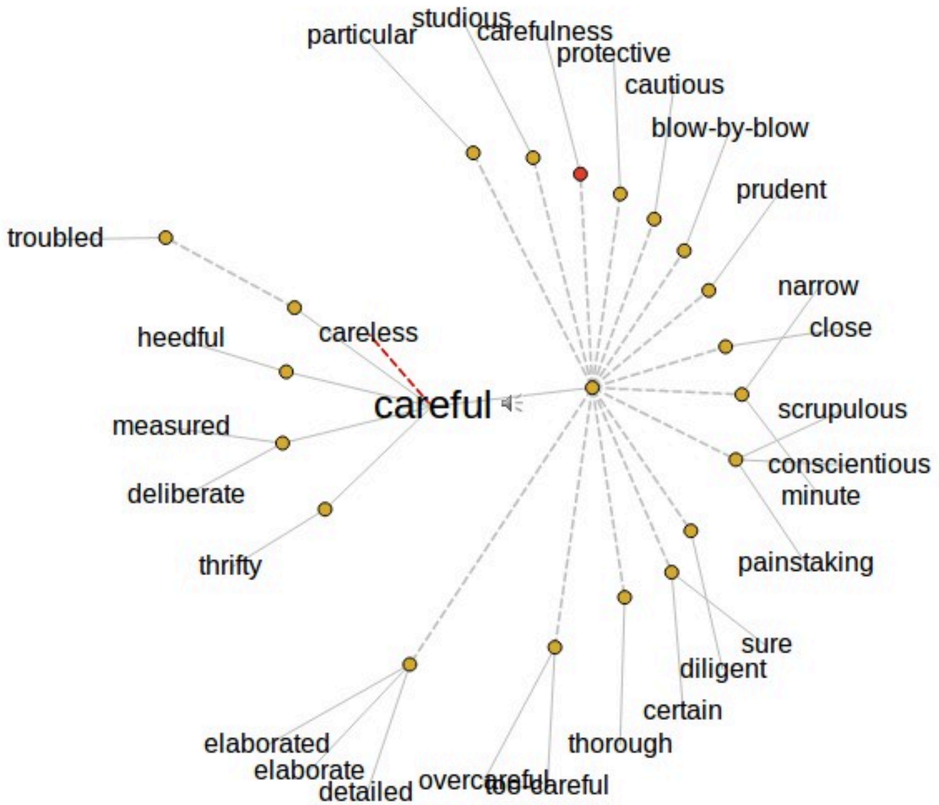
pleasantness intensity degree of control

	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

Annotated resources for lexical semantics

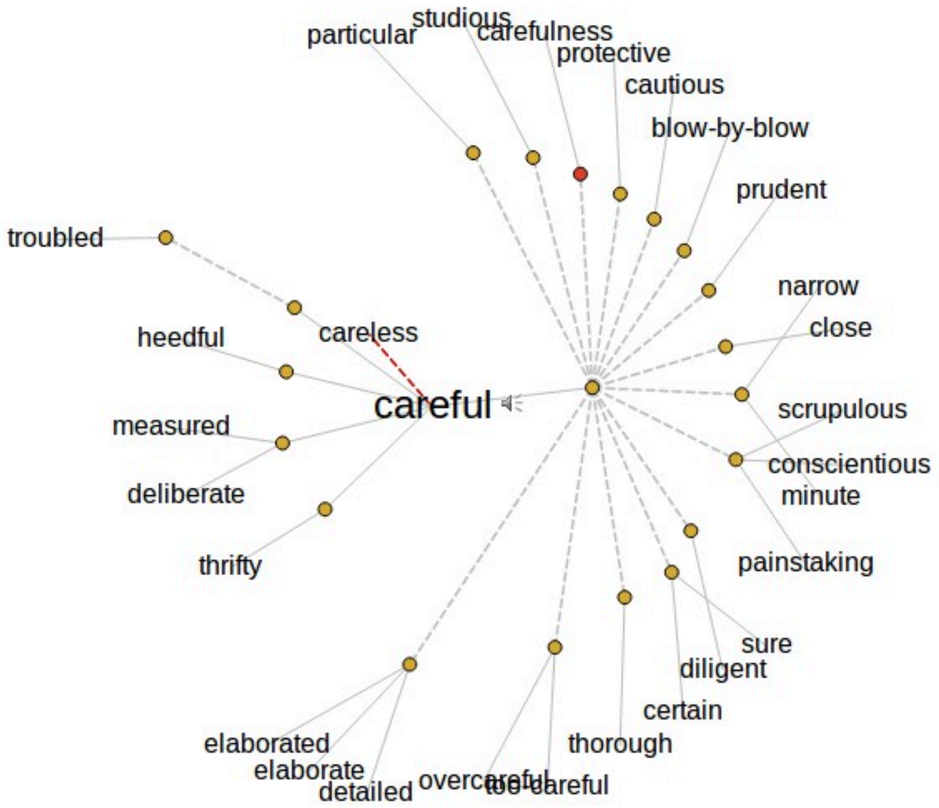
Annotated resources for lexical semantics

- WordNet: https://wordnet.princeton.edu/



Annotated resources for lexical semantics

- WordNet: https://wordnet.princeton.edu/



WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: bank Search WordNet

Display Options: (Select option to change) Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **bank** (sloping l *pulled the canoe up watched the currents*
- S: (n) depository fina company (a financial money into lending a *holds the mortgage*
- S: (n) **bank** (a long ri
- S: (n) **bank** (an arran *operated a bank of s*
- S: (n) **bank** (a supply emergencies))
- S: (n) **bank** (the fund gambling games) *"he*
- S: (n) **bank**, cant, car is higher than the ins
- S: (n) savings bank, slot in the top) for ke
- S: (n) **bank**, bank bui transacted) *"the bank*
- S: (n) **bank** (a flight axis (especially in tur

Verb

- S: (v) **bank** (tip latera
- S: (v) **bank** (enclose
- S: (v) **bank** (do busin *do you bank in this t*
- S: (v) **bank** (act as th
- S: (v) **bank** (be in the
- S: (v) deposit, **bank** (*every month"*
- S: (v) **bank** (cover wit
- S: (v) count, bet, dep faith or confidence ir *your friends for supp times of crisis"*

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

How to represent a word?

Vector space model

How to represent a word?

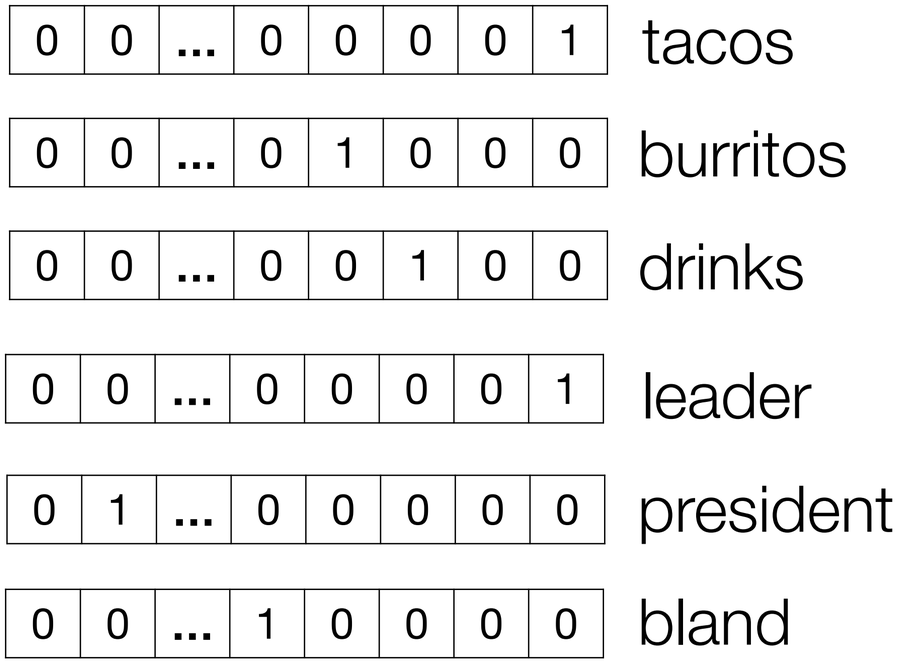
Vector space model

- Until the ~2010s, in NLP words == atomic symbols

How to represent a word?

Vector space model

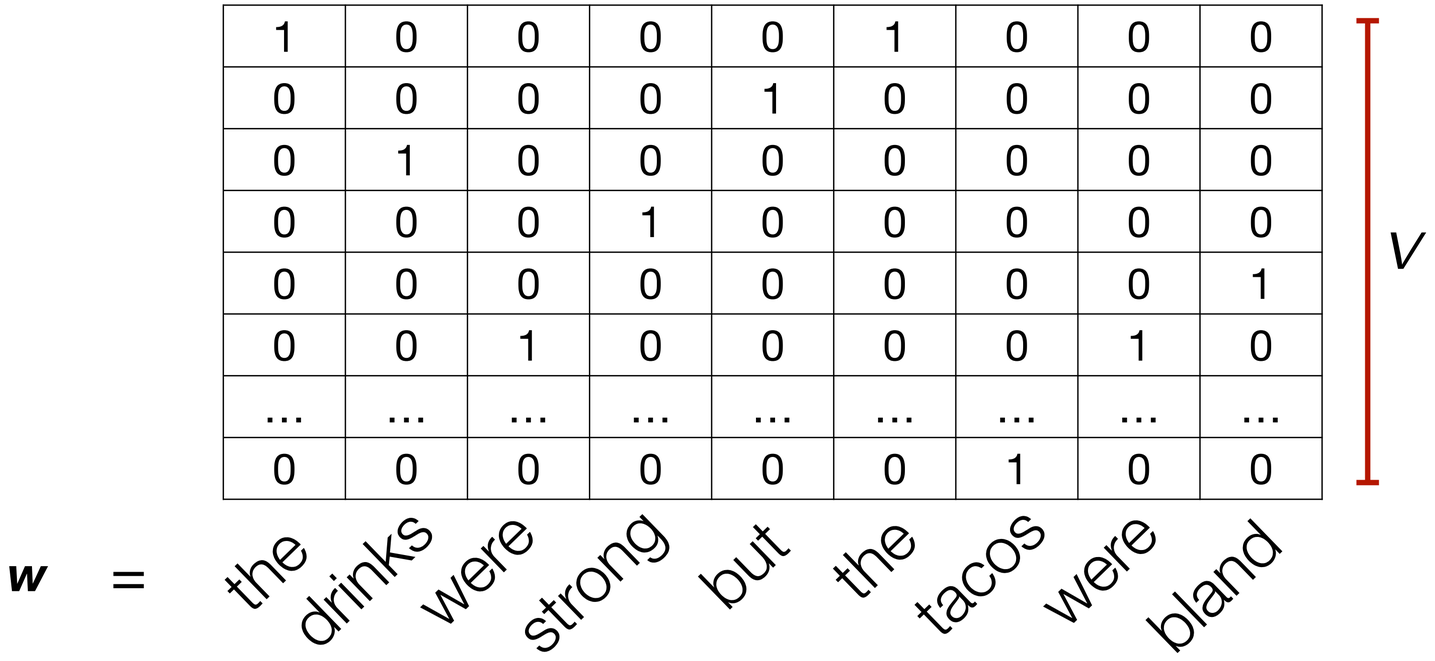
- Until the ~2010s, in NLP words == atomic symbols
- **One-hot** representations in vector space:

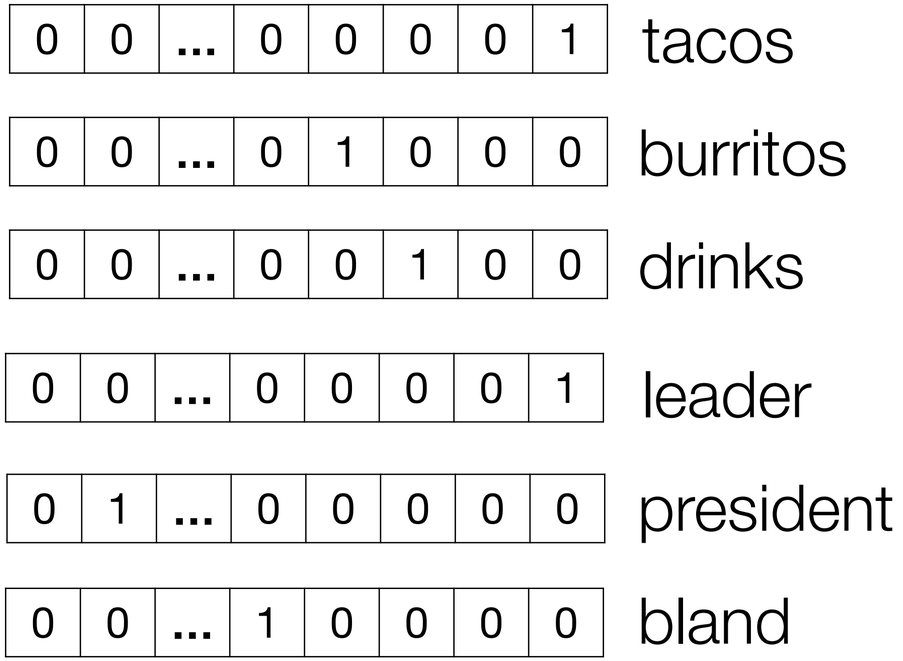


How to represent a word?

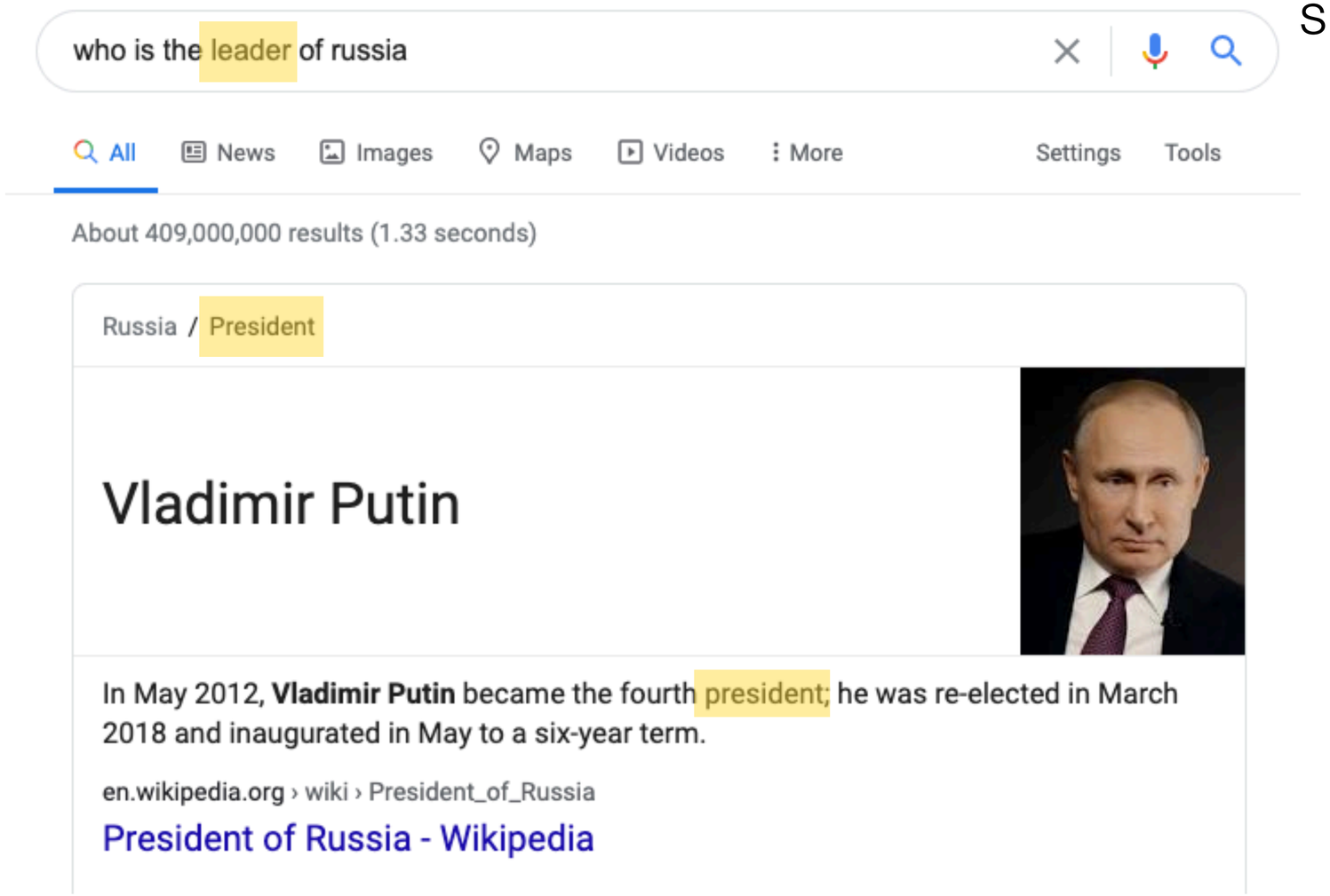
Vector space model

- Until the ~2010s, in NLP words == atomic symbols
- **One-hot** representations in vector space:





How to represent a word?

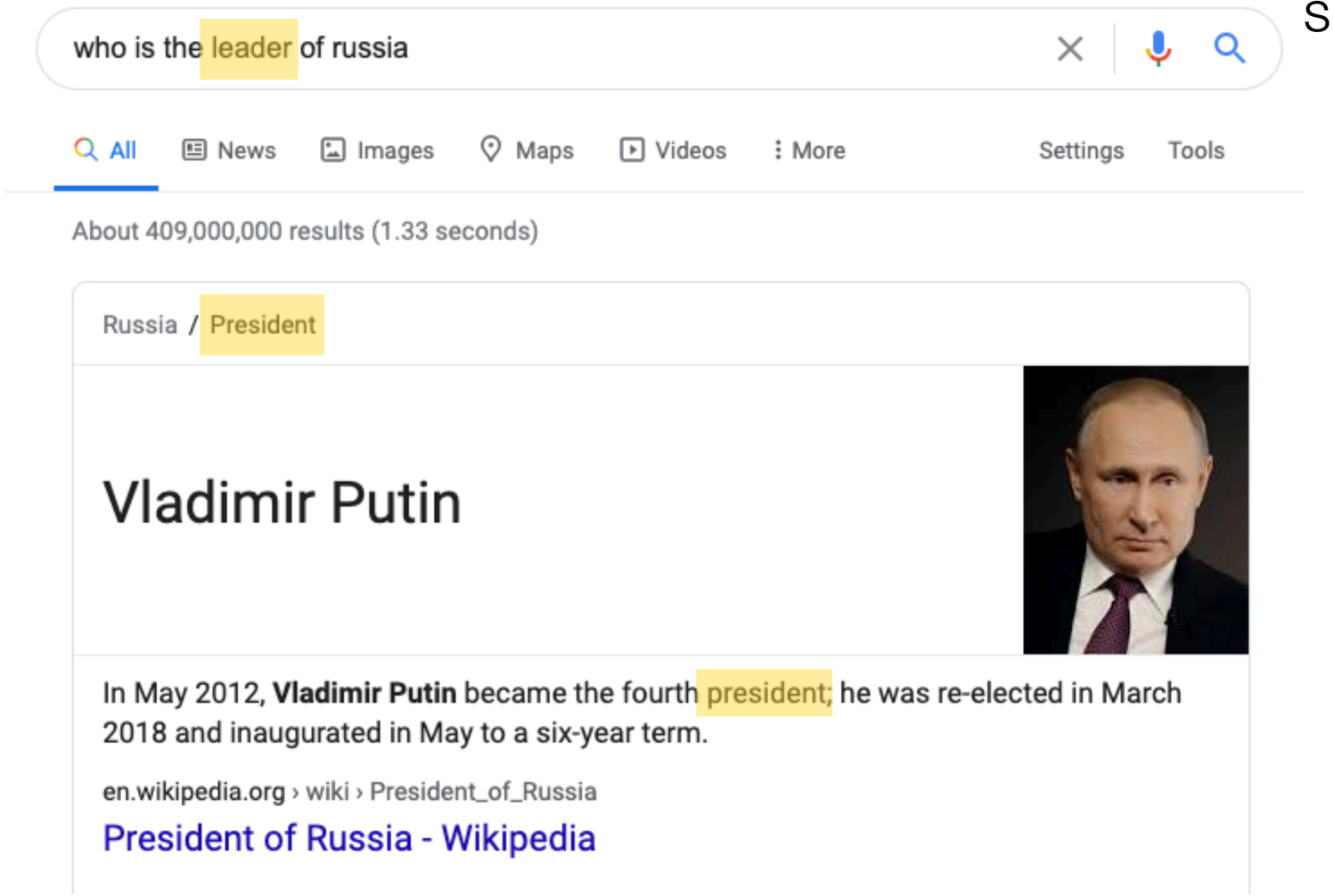


S

0	0	...	0	0	0	0	1	tacos
0	0	...	0	1	0	0	0	burritos
0	0	...	0	0	1	0	0	drinks
0	0	...	0	0	0	1	0	leader
0	1	...	0	0	0	0	0	president
0	0	...	1	0	0	0	0	bland

w

How to represent a word?

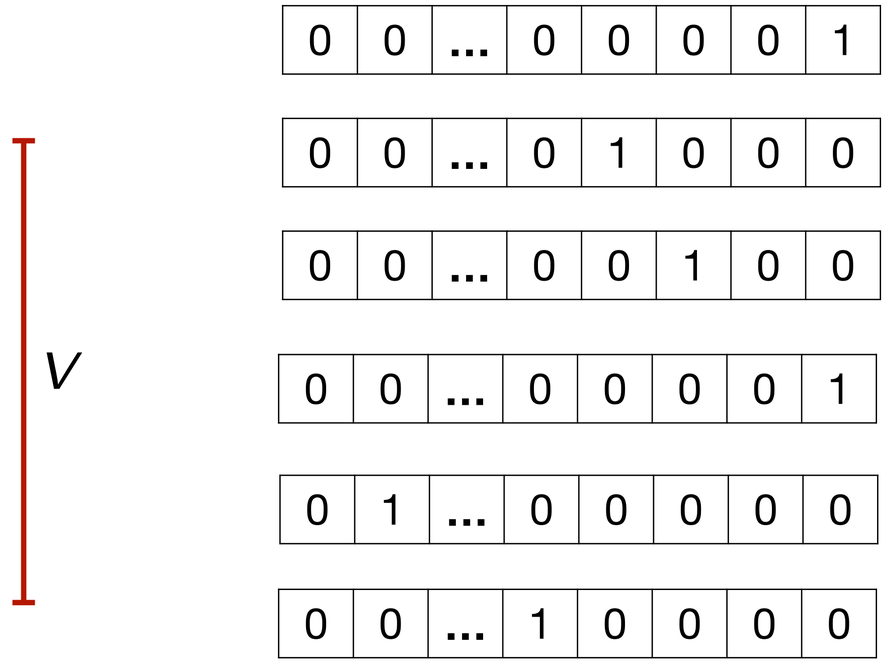


S

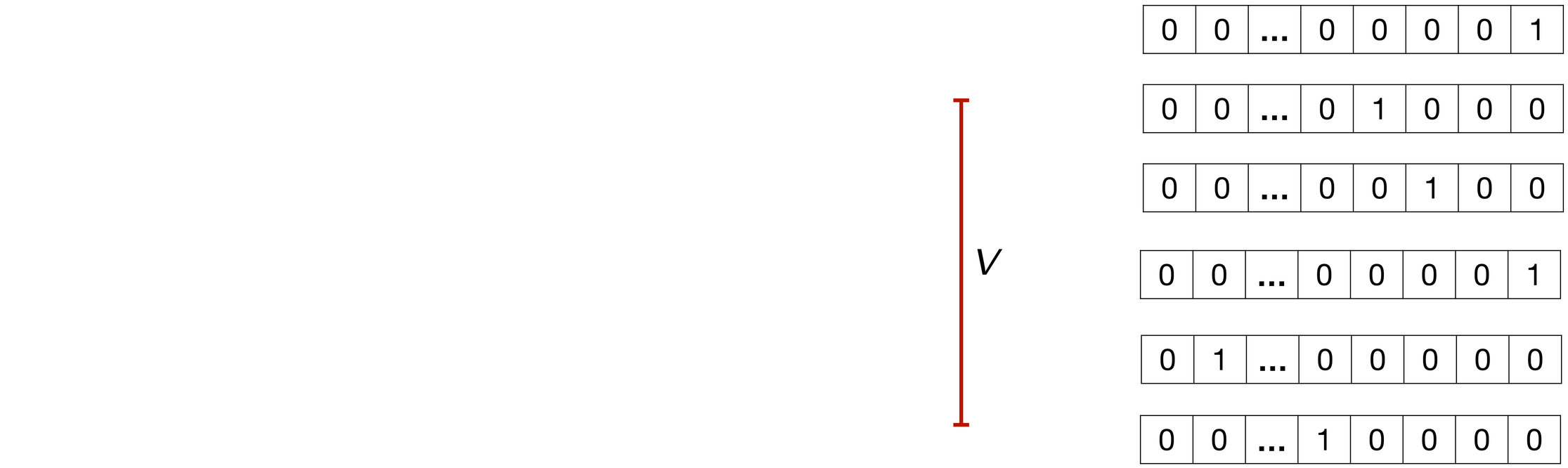
0	0	...	0	0	0	0	1	tacos
0	0	...	0	1	0	0	0	burritos
0	0	...	0	0	1	0	0	drinks
0	0	...	0	0	0	1	0	leader
0	1	...	0	0	0	0	0	president
0	0	...	1	0	0	0	0	bland

w

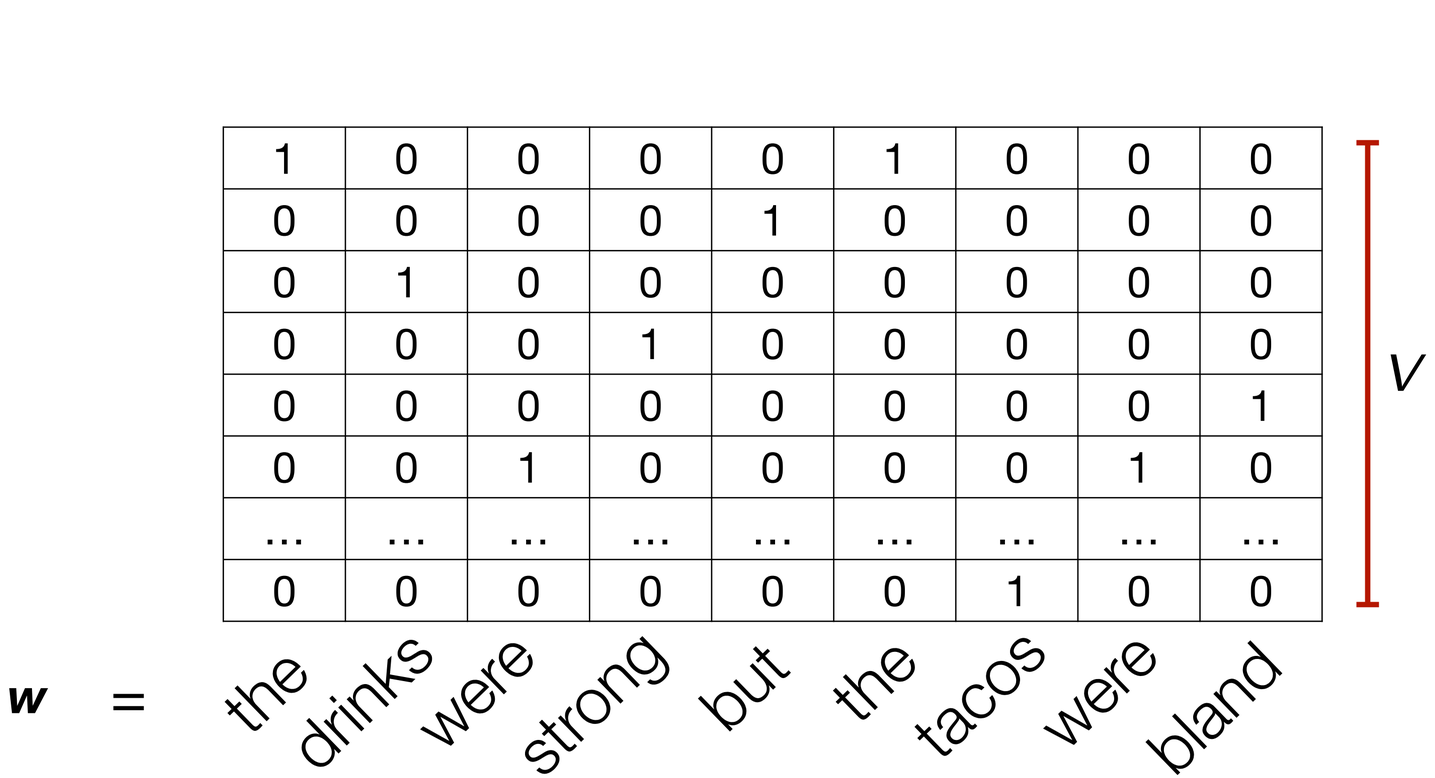
How to represent a word?



How to represent a word?



How to represent a word?

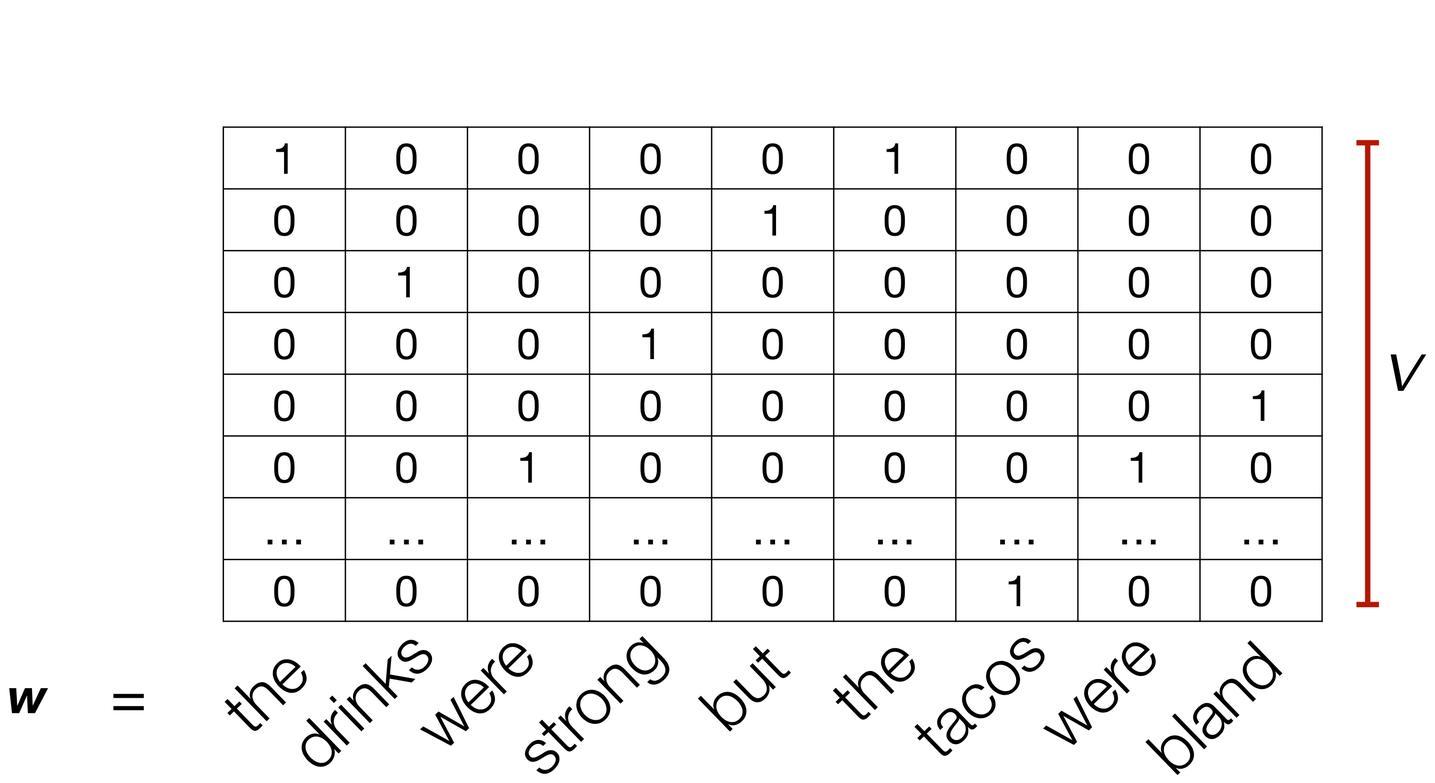


1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	0
…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	1	0	0

0	0	…	0	0	0	0	1
0	0	…	0	1	0	0	0
0	0	…	0	0	1	0	0
0	0	…	0	0	0	0	1
0	1	…	0	0	0	0	0
0	0	…	1	0	0	0	0

How to represent a word?

- Until the ~2010s, in NLP words == atomic symbols.

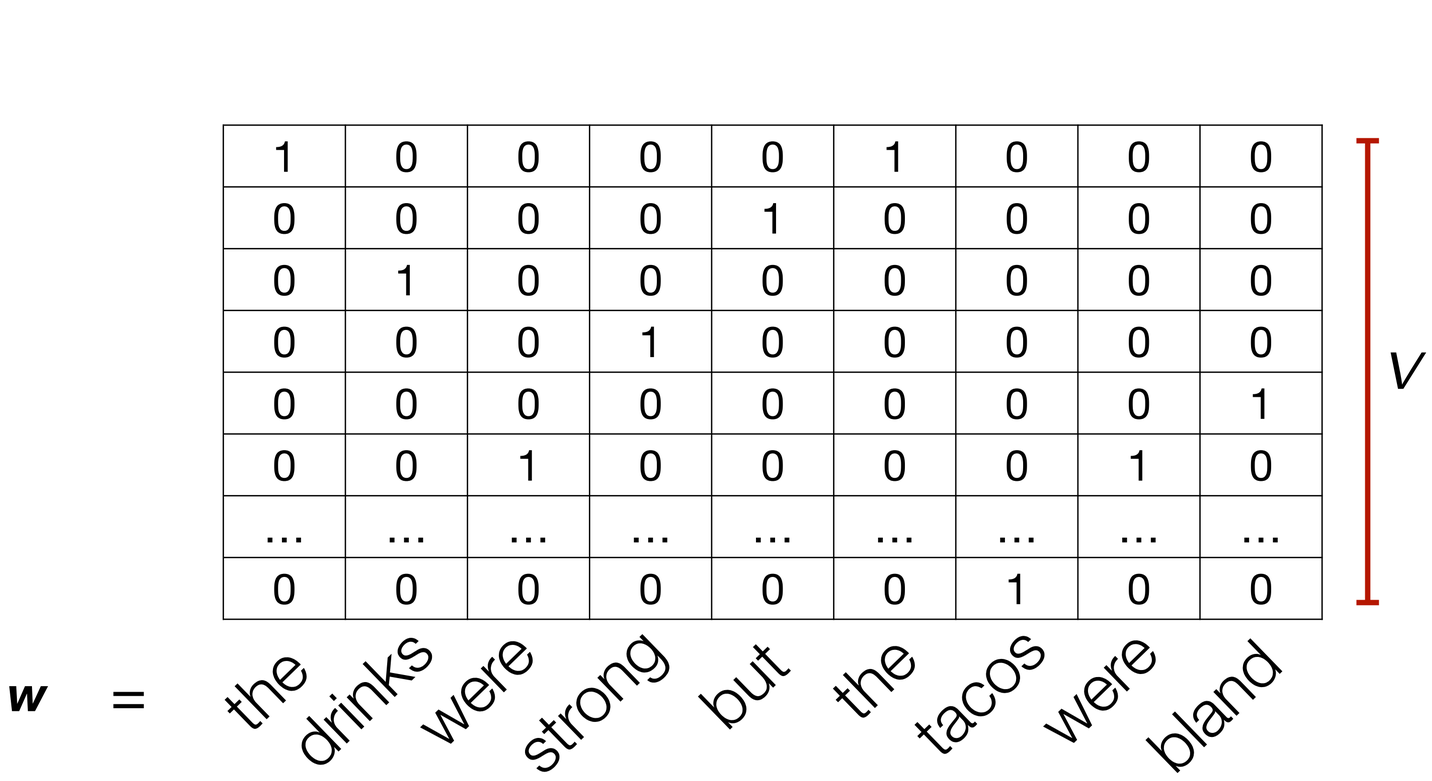


1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	0
…	…	…	…	…	…	…	…	…
0	0	0	0	0	0	1	0	0

0	0	…	0	0	0	0	1
0	0	…	0	1	0	0	0
0	0	…	0	0	1	0	0
0	0	…	0	0	0	0	1
0	1	…	0	0	0	0	0
0	0	…	1	0	0	0	0

How to represent a word?

- Until the ~2010s, in NLP words == atomic symbols.
- **One-hot** representations in vector space.



1	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...
0	0	0	0	0	0	1	0	0

0	0	...	0	0	0	0	1
0	0	...	0	1	0	0	0
0	0	...	0	0	1	0	0
0	0	...	0	0	0	0	1
0	1	...	0	0	0	0	0
0	0	...	1	0	0	0	0

How to represent a word?

- Until the ~2010s, in NLP words == atomic symbols.
- **One-hot** representations in vector space.
- But *leader* and *president* are orthogonal!

	the	drinks	were	strong	but	the	tacos	were	bland
	1	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	0	0	0
	0	1	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	0
	0	0	0	0	0	0	0	0	1
	0	0	1	0	0	0	0	1	0
	...	...	...	...	...	...	...	...	...
	0	0	0	0	0	0	1	0	0

w =

V

0	0	...	0	0	0	0	1	tacos
0	0	...	0	1	0	0	0	burritos
0	0	...	0	0	1	0	0	drinks
0	0	...	0	0	0	0	1	leader
0	1	...	0	0	0	0	0	president
0	0	...	1	0	0	0	0	bland

How to represent a word?

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don’t have tezgüino before you drive.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don’t have tezgüino before you drive.
4. We make tezgüino out of corn.

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don't have tezgüino before you drive.
4. We make tezgüino out of corn.

term	context 1	2	3	4
tezgüino	1	1	1	1
loud	0	0	0	0
motor oil	1	0	0	1
tortillas	0	1	0	1
choices	0	1	0	0
wine	1	1	1	0

How to represent a word?

- Solution: encode word *similarity* (not just identity) in word representations.
- How to encode similarity?
- Consider encountering a new word: tezgüino.

1. A bottle of tezgüino is on the table.
2. Everybody likes tezgüino.
3. Don't have tezgüino before you drive.
4. We make tezgüino out of corn.

term	context 1	2	3	4	similarity?
tezgüino	1	1	1	1	
loud	0	0	0	0	0
motor oil	1	0	0	1	2
tortillas	0	1	0	1	2
choices	0	1	0	0	1
wine	1	1	1	0	3

Distributional hypothesis

Distributional hypothesis

- These representations encode **distributional** properties of each word.

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

> “The meaning of a word it its use in the language.” [Wittgenstein 1943]

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

"The meaning of a word it its use in the language." [Wittgenstein 1943]

"If A and B have almost identical environments we say that they are synonyms." [Harris 1954]

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

"The meaning of a word it its use in the language." [Wittgenstein 1943]

"If A and B have almost identical environments we say that they are synonyms." [Harris 1954]

"You shall know a word by the company it keeps." [Firth 1957]

Distributional hypothesis

- These representations encode **distributional** properties of each word.
- The **distributional hypothesis**: words with similar meaning are used in similar contexts.

How to encode context?

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.

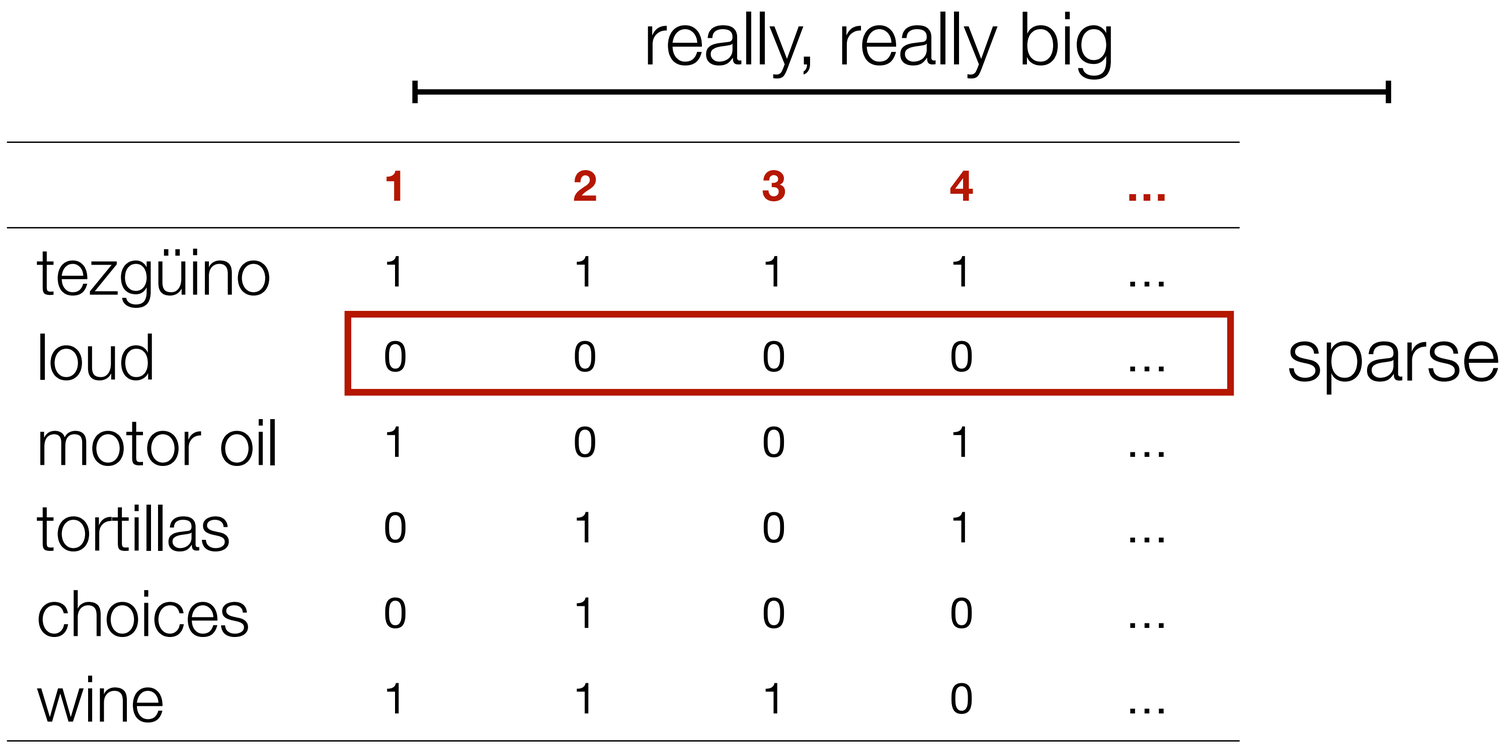
really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

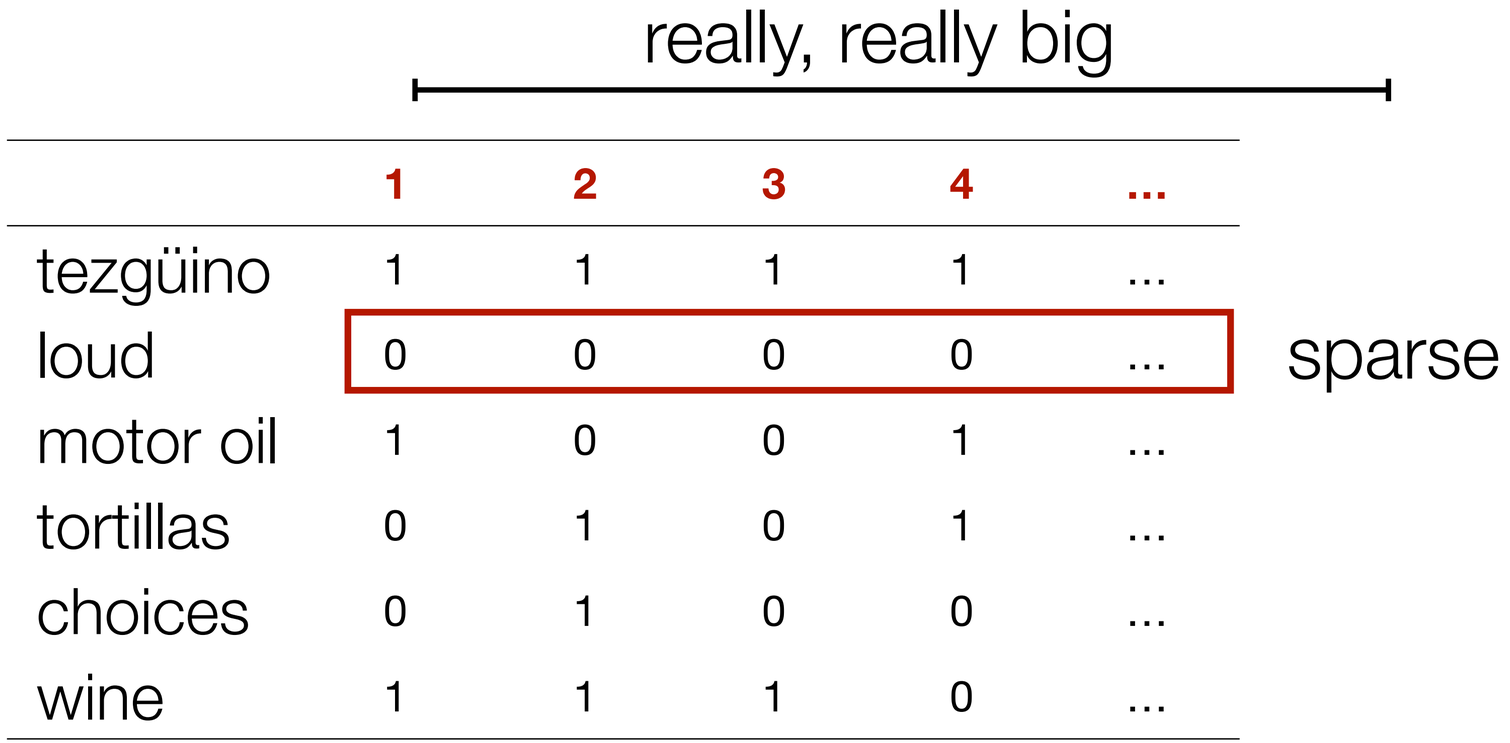
How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*



How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:



How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization
  - Brown clusters

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization
  - Brown clusters
  - tf-idf

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization
  - Brown clusters
  - tf-idf
  - word2vec

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization
  - Brown clusters
  - **tf-idf**
  - **word2vec**

really, really big

	1	2	3	4	…
tezgüino	1	1	1	1	…
loud	0	0	0	0	…
motor oil	1	0	0	1	…
tortillas	0	1	0	1	…
choices	0	1	0	0	…
wine	1	1	1	0	…

sparse

Encoding context with tf-idf

Encoding context with tf-idf

- Consider a matrix of word counts across documents: **term-document matrix**

Encoding context with tf-idf

- Consider a matrix of word counts across documents: **term-document matrix**

	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

Encoding context with tf-idf

- Consider a matrix of word counts across documents: **term-document matrix**

	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

bag-of-words
(document representation)

Encoding context with tf-idf

- Consider a matrix of word counts across documents: **term-document matrix**

	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

word vector

bag-of-words
(document representation)

Encoding context with tf-idf

- Consider a matrix of word counts across documents: **term-document matrix**
- Same problem as term-context matrix: useless signal from *the*, *they*, *and*

	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

word vector

bag-of-words
(document representation)

Encoding context with tf-idf

Encoding context with tf-idf

- Solution: **tf-idf** incorporates two terms that capture these conflicting constraints:

Encoding context with tf-idf

- Solution: **tf-idf** incorporates two terms that capture these conflicting constraints:
- **Term frequency** (tf): frequency of the word *t* in the document *d*

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1)$$

Encoding context with tf-idf

- Solution: **tf-idf** incorporates two terms that capture these conflicting constraints:
- **Term frequency** (tf): frequency of the word *t* in the document *d*

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1)$$

- **Document frequency** (df): number of documents that a term occurs in.
  Inverse document frequency (idf):

$$\text{idf}_t = \log \frac{N}{df_t}$$

  where *N* is the number of documents in the corpus.

Encoding context with tf-idf

- Solution: **tf-idf** incorporates two terms that capture these conflicting constraints:
- **Term frequency** (tf): frequency of the word *t* in the document *d*

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1)$$

- **Document frequency** (df): number of documents that a term occurs in.
  Inverse document frequency (idf):

$$\text{idf}_t = \log \frac{N}{df_t}$$

higher for terms that occur in fewer documents

where *N* is the number of documents in the corpus.

Encoding context with tf-idf

- Solution: **tf-idf** incorporates two terms that capture these conflicting constraints:
- **Term frequency** (tf): frequency of the word *t* in the document *d*

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1)$$

- **Document frequency** (df): number of documents that a term occurs in.
  Inverse document frequency (idf):

$$\text{idf}_t = \log \frac{N}{df_t}$$

higher for terms that occur in fewer documents

  where *N* is the number of documents in the corpus.

- **tf-idf** combines these two terms:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

Encoding context with tf-idf

- **tf-idf** combines term frequency and inverse document frequency:

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1) \qquad \text{idf}_t = \log \frac{N}{df_t} \qquad \text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

Encoding context with tf-idf

- **tf-idf** combines term frequency and inverse document frequency:

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1) \qquad \text{idf}_t = \log \frac{N}{df_t} \qquad \text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

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

Encoding context with tf-idf

- **tf-idf** combines term frequency and inverse document frequency:

$$\text{tf}_{t,d} = \log(\text{count}(t,d)+1) \qquad \text{idf}_t = \log\frac{N}{df_t} \qquad \text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

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

	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

Encoding context with tf-idf

- **tf-idf** combines term frequency and inverse document frequency:

$$\text{tf}_{t,d} = \log(\text{count}(t,d)+1) \qquad \text{idf}_t = \log\frac{N}{df_t} \qquad \text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$$

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

	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

tf-idf today

tf-idf today

- tf-idf was designed for and still excels at **document retrieval**

tf-idf today

- tf-idf was designed for and still excels at **document retrieval**
- The **BM25** model (very similar to tf-idf) is still a strong document retrieval baseline!

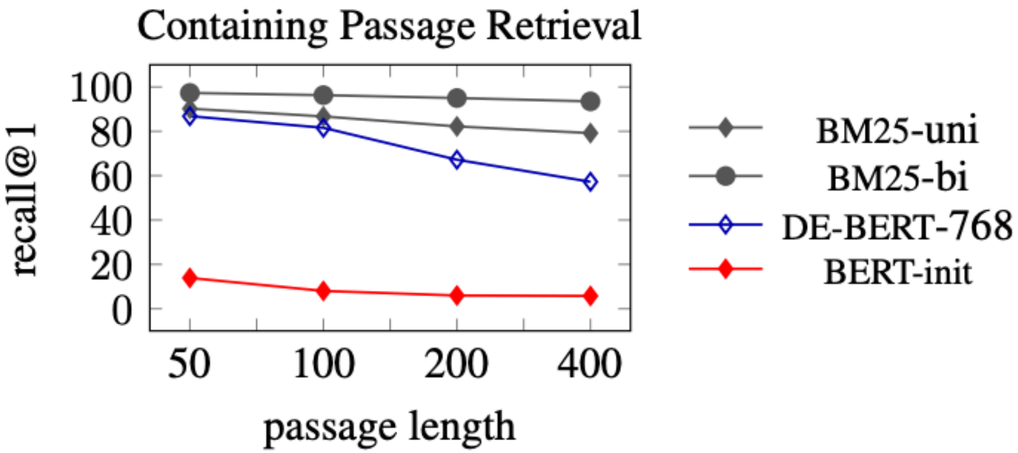
$$\text{BM25}(d, \mathbf{q}) = \sum_{q_i \in \mathbf{q}} \text{idf}_{q_i} \cdot \frac{tf_{q_i,d} \cdot (k_1 + 1)}{tf_{q_i,d} + k_1(1 - b + b\frac{|D|}{avg|D|})}$$

tf-idf today

- tf-idf was designed for and still excels at **document retrieval**
- The **BM25** model (very similar to tf-idf) is still a strong document retrieval baseline!

$$\text{BM25}(d, \mathbf{q}) = \sum_{q_i \in \mathbf{q}} \text{idf}_{q_i} \cdot \frac{tf_{q_i,d} \cdot (k_1 + 1)}{tf_{q_i,d} + k_1(1 - b + b\frac{|D|}{avg|D|})}$$

Recent history in NLP might suggest that learned dense representations should always outperform sparse features, but this is not necessarily true: as shown in Figure 1, the BM25 model (Robertson et al., 2009) can outperform a dual encoder based on BERT, particularly on longer documents (See § 7).



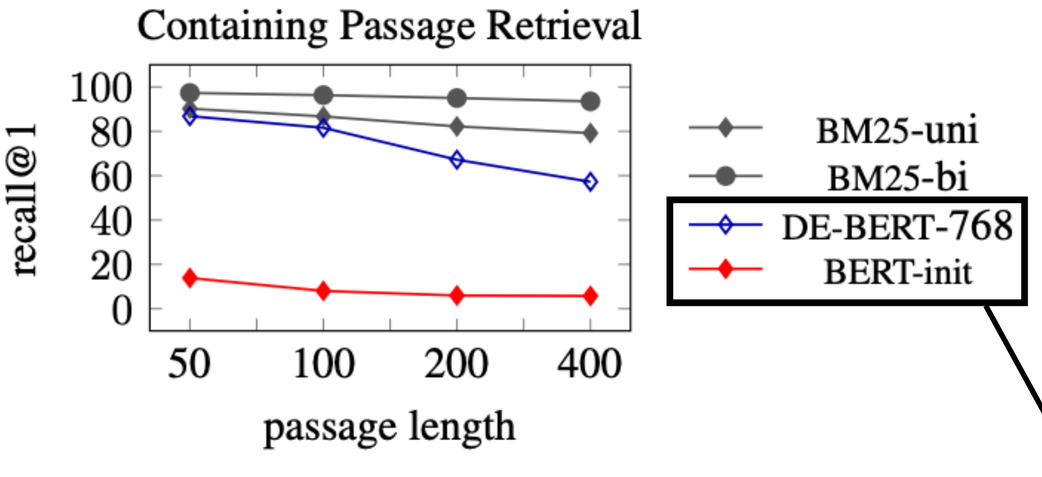
Yi Luan, Jacob Eisenstein, Kristina Toutanova, Michael Collins. Sparse, dense and attentional representations for text retreival. arXiv preprint 2005.00181, 2020.

tf-idf today

- tf-idf was designed for and still excels at **document retrieval**
- The **BM25** model (very similar to tf-idf) is still a strong document retrieval baseline!

$$\text{BM25}(d, \mathbf{q}) = \sum_{q_i \in \mathbf{q}} \text{idf}_{q_i} \cdot \frac{tf_{q_i,d} \cdot (k_1 + 1)}{tf_{q_i,d} + k_1(1 - b + b\frac{|D|}{avg|D|})}$$

Recent history in NLP might suggest that learned dense representations should always outperform sparse features, but this is not necessarily true: as shown in Figure 1, the BM25 model (Robertson et al., 2009) can outperform a dual encoder based on BERT, particularly on longer documents (See § 7).



Fancy, computationally expensive neural network models!

Yi Luan, Jacob Eisenstein, Kristina Toutanova, Michael Collins. Sparse, dense and attentional representations for text retreival. arXiv preprint 2005.00181, 2020.

Dimensionality reduction

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
    - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy
  - In practice, they just work better [Baroni et al. 2014].

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy
  - In practice, they just work better [Baroni et al. 2014].
- Goal:

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy
  - In practice, they just work better [Baroni et al. 2014].
- Goal:
  - Good similarity measure for words and documents

Dimensionality reduction

- Wikipedia: ~29 million English documents. Vocab: ~1 million words.
  - High dimensionality term-document matrix; Highly sparse.
- Sparse vs. dense vectors:
  - Short vectors often easier to use as features in a classifier (fewer parameters).
  - Dense vectors may generalize better than storing explicit counts.
  - May better capture synonymy
  - In practice, they just work better [Baroni et al. 2014].
- Goal:
  - Good similarity measure for words and documents
  - Dense representation

Measuring word vector similarity

Measuring word vector similarity

- **Dot product** (inner product):

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

Measuring word vector similarity

- **Dot product** (inner product):

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

Measuring word vector similarity

- **Dot product** (inner product):

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

+ + − −

Measuring word vector similarity

- **Dot product** (inner product):

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

$+$ $+$ $-$

$+$ $+$ $-$ $-$ $+$ $-$

Measuring word vector similarity

- **Dot product** (inner product):

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

$+$ $+$ $-$

$+\ +$ $-\ -$ $+\ -$

favors frequent words.

Measuring word vector similarity

- **Dot product** (inner product):

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

  (+ + − − + −)

  favors frequent words.

- Normalize by vector norm: **cosine similarity:**

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

Measuring word vector similarity

- **Dot product** (inner product):

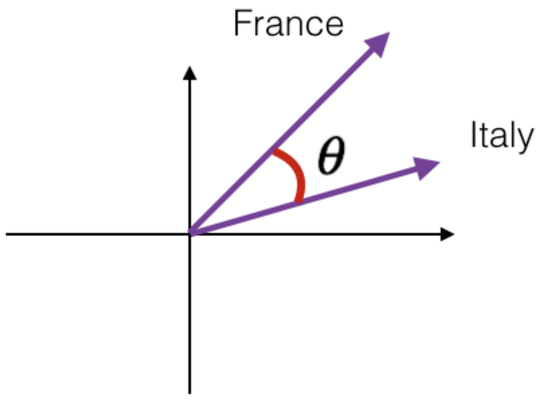
$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

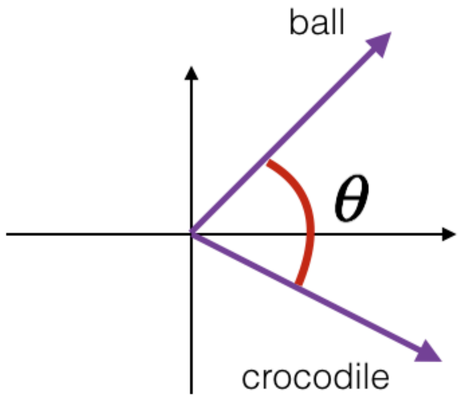
(+ + − − + −)

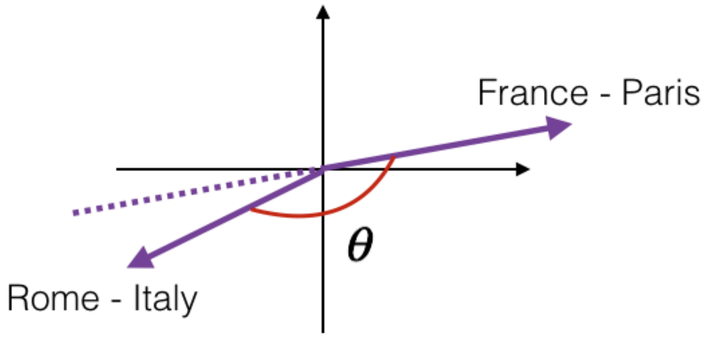
favors frequent words.

- Normalize by vector norm: **cosine similarity:**

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







Measuring word vector similarity

- **Dot product** (inner product):

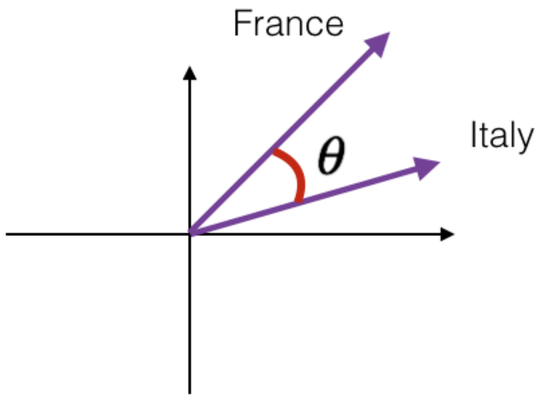
$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

(+ + − − + −)

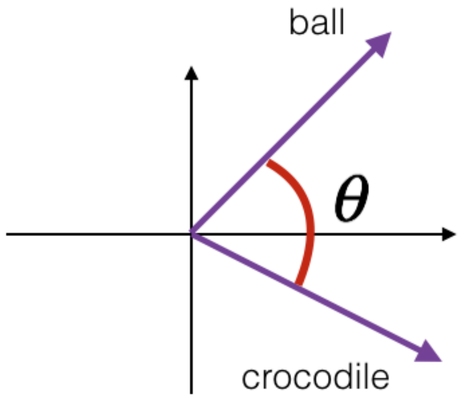
favors frequent words.

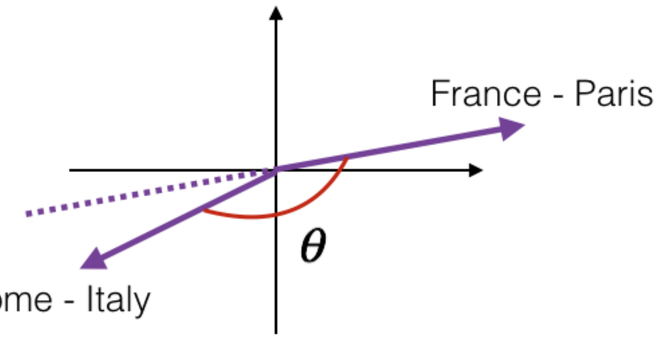
- Normalize by vector norm: **cosine similarity:**

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



$\cos(\mathbf{v}, \mathbf{w}) \approx 1$





Measuring word vector similarity

- **Dot product** (inner product):

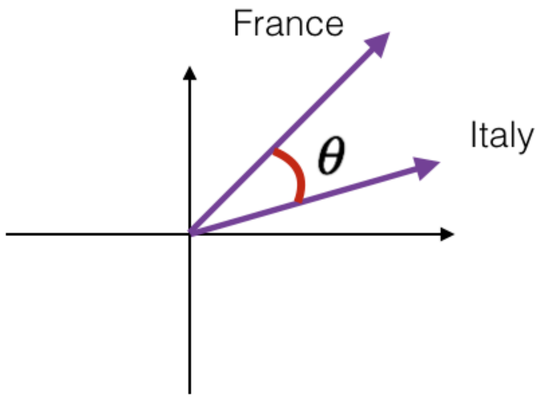
$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

(+ + − − + −)

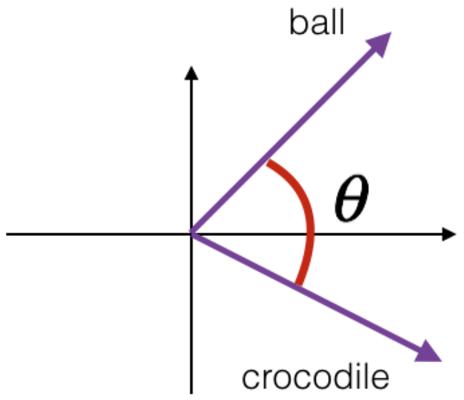
favors frequent words.

- Normalize by vector norm: **cosine similarity:**

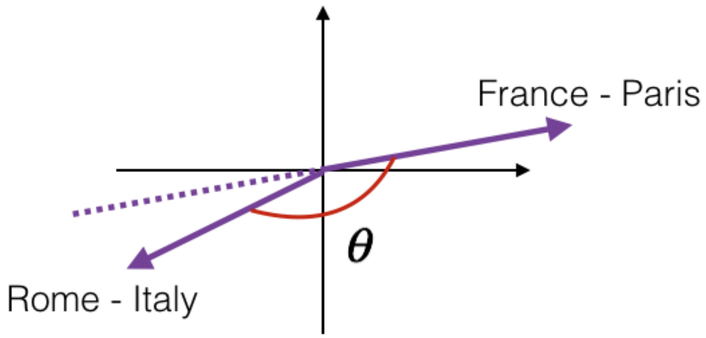
$$cosine(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$



$\cos(\mathbf{v}, \mathbf{w}) \approx 1$



$\cos(\mathbf{v}, \mathbf{w}) \approx 0$



Measuring word vector similarity

- **Dot product** (inner product):

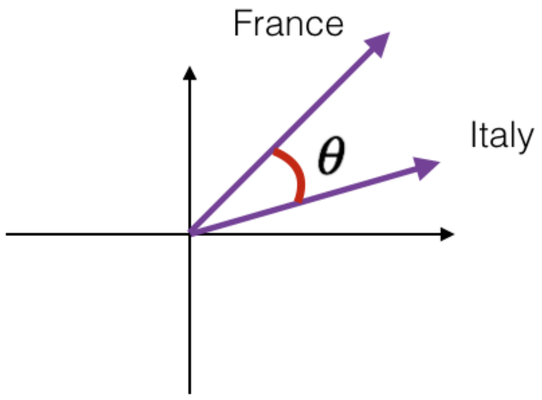
$$\mathbf{v} \cdot \mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N$$

(+ + − − + −)

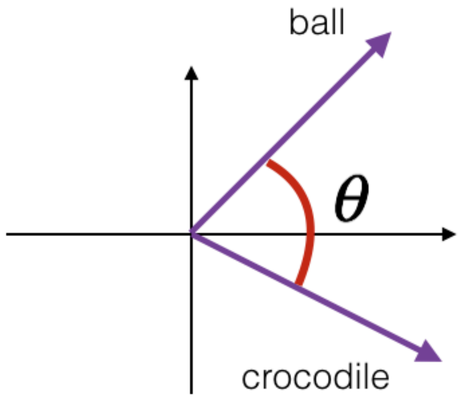
favors frequent words.

- Normalize by vector norm: **cosine similarity:**

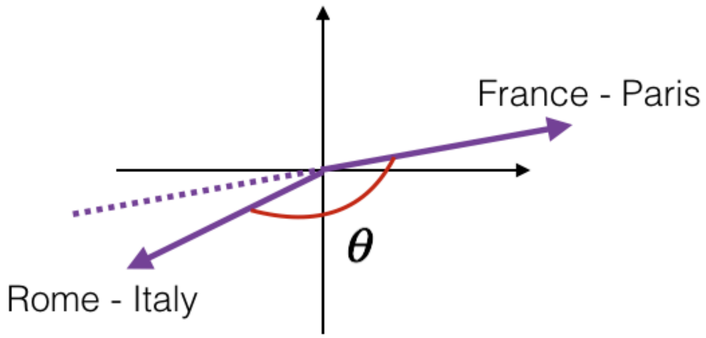
$$cosine(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$



$\cos(\mathbf{v}, \mathbf{w}) \approx 1$



$\cos(\mathbf{v}, \mathbf{w}) \approx 0$



$\cos(\mathbf{v}, \mathbf{w}) \approx -1$

How to encode context?

- This encoding could work, but it's not practical to enumerate all possible contexts.
- Also, many entries not very discriminative: *the*, *it*, *they*
- Approaches for encoding words as vectors:
  - Matrix factorization
  - Brown clusters
  - tf-idf
  - **word2vec**

$$\overrightarrow{king} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{queen}$$

word2vec

word2vec

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

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don’t actually care about performing this task, but we’ll take the learned classifier weights as the word embeddings.

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don't actually care about performing this task, but we'll take the learned classifier weights as the word embeddings.
- Training is **self-supervised:** no annotated data required, just raw text!

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don't actually care about performing this task, but we'll take the learned classifier weights as the word embeddings.
- Training is **self-supervised:** no annotated data required, just raw text!
- Two algorithms:

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don't actually care about performing this task, but we'll take the learned classifier weights as the word embeddings.
- Training is **self-supervised:** no annotated data required, just raw text!
- Two algorithms:
  - Context bag-of-words (CBOW): predict current word using context

$$P(w_t \mid w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k})$$

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don't actually care about performing this task, but we'll take the learned classifier weights as the word embeddings.
- Training is **self-supervised:** no annotated data required, just raw text!
- Two algorithms:
  - Context bag-of-words (CBOW): predict current word using context

$$P(w_t \mid w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k})$$

  - Skip-gram: predict each context word using current word

$$P(w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k} \mid w_t)$$

word2vec

- Instead of counting how often each word *w* occurs near *apricot*
- Train a classifier on a binary prediction task: Is *w* likely to show up near *apricot*?
- Don't actually care about performing this task, but we'll take the learned classifier weights as the word embeddings.
- Training is **self-supervised:** no annotated data required, just raw text!
- Two algorithms:
  - Context bag-of-words (CBOW): predict current word using context

$$P(w_t \mid w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k})$$

  - Skip-gram: predict each context word using current word

$$P(w_{t+1}, \ldots, w_{t+k}, w_{t-1}, \ldots, w_{t-k} \mid w_t)$$

skip-gram

skip-gram

… that Europe needs unified **banking** regulation to replace the hodgepodge …

skip-gram

… that Europe needs unified **banking** regulation to replace the hodgepodge …

skip-gram

… that Europe needs unified **banking** regulation to replace the hodgepodge …

- Objective: Maximize probability of any context word given center word:

$$L = \frac{1}{T} \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

skip-gram

… that Europe needs unified **banking** regulation to replace the hodgepodge …

- Objective: Maximize probability of any context word given center word:

$$L = \frac{1}{T} \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} \mid w_t; \theta)$$

- Minimize negative log probability:

$$= -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

skip-gram

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

all tokens in the corpus

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

all tokens in the corpus

all context for token *t*

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

softmax

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

dot product (similarity) between outside and center word vectors

softmax

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V}\exp(u_i^T v_c)}$$

dot product (similarity) between outside and center word vectors

softmax

o = index of outside (context) word

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m \\ j\neq 0}} \log P(w_{t+j} \mid w_t; \theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

dot product (similarity) between outside and center word vectors

softmax

o = index of outside (context) word

c = index of center word (w_t)

skip-gram

- Minimize negative log probability:

$$= -\frac{1}{T}\sum_{t=1}^{T}\sum_{\substack{-m\leq j\leq m\\ j\neq 0}} \log P(w_{t+j} \mid w_t;\theta)$$

all tokens in the corpus

all context for token *t*

- Again, using logistic regression:

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V}\exp(u_i^T v_c)}$$

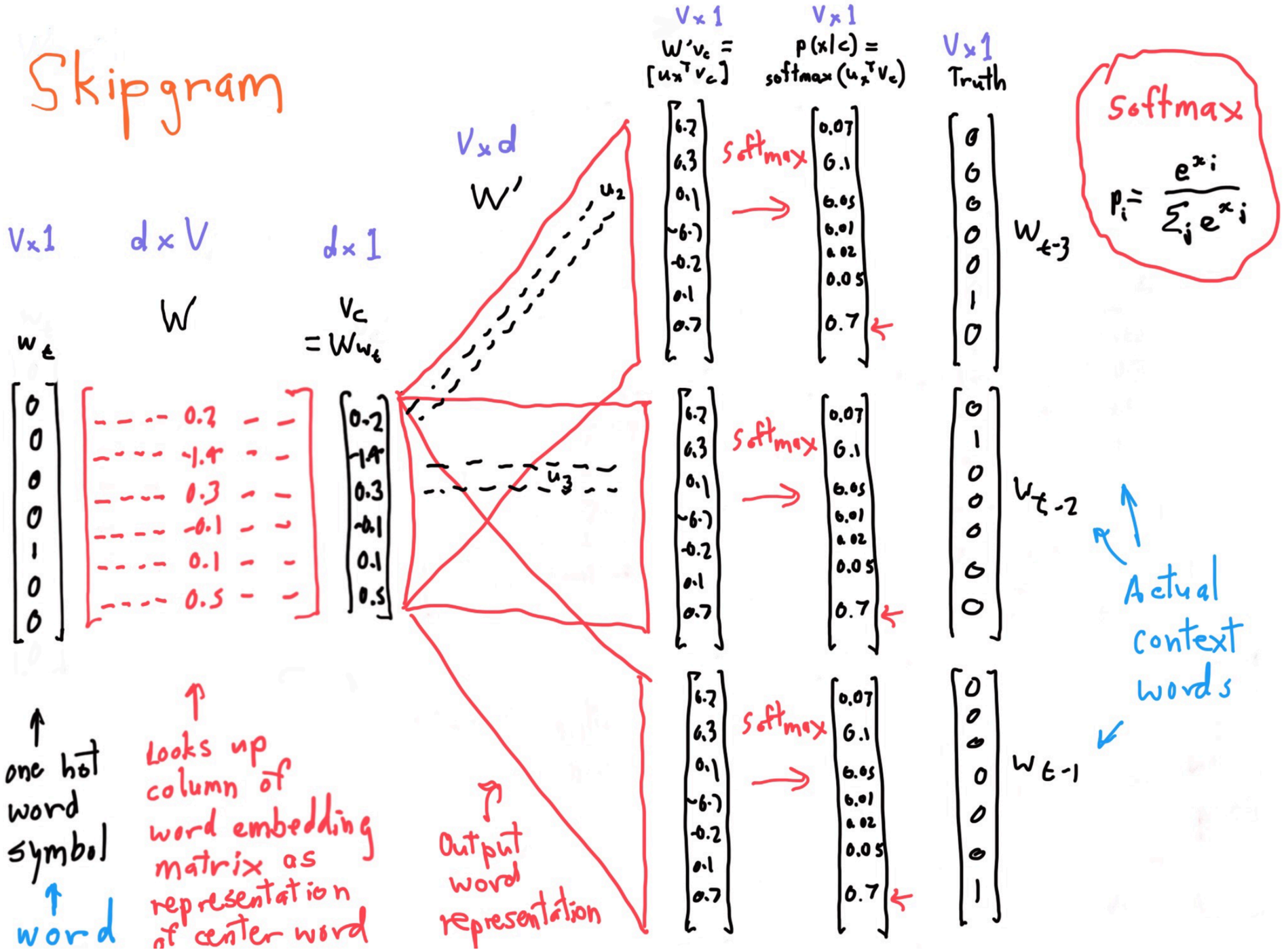
dot product (similarity) between outside and center word vectors

softmax

o = index of outside (context) word

c = index of center word (w_t)

V = vocab size



slide from: http://web.stanford.edu/class/cs224n/

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

- V = 50K — 30M

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

- V = 50K — 30M $O(VK)$

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$
- Two algorithms:

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

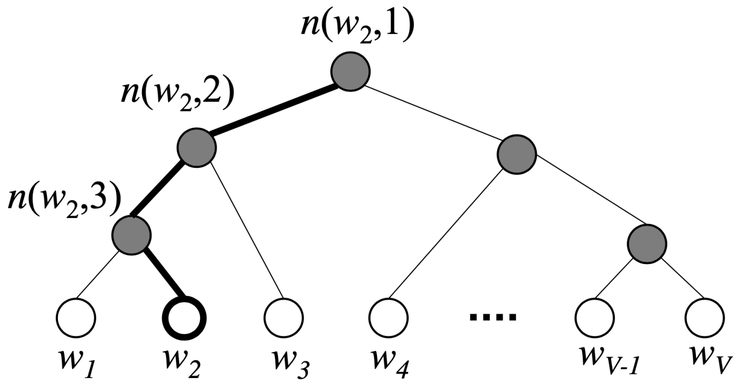
- V = 50K — 30M $O(VK)$
- Two algorithms:
  - Hierarchical softmax:

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$
- Two algorithms:
  - Hierarchical softmax:

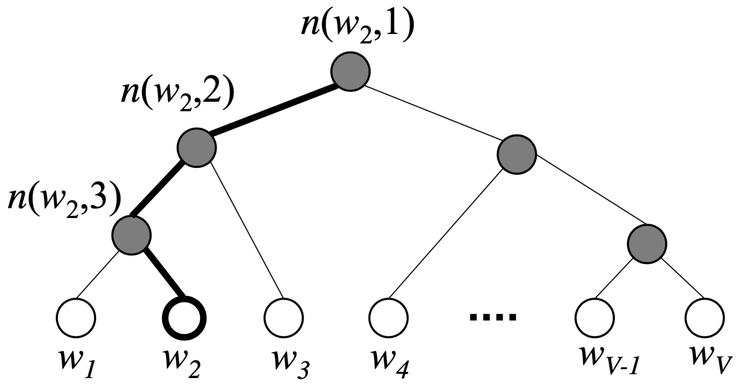


skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$
- Two algorithms:
  - Hierarchical softmax: $O(\log V)$

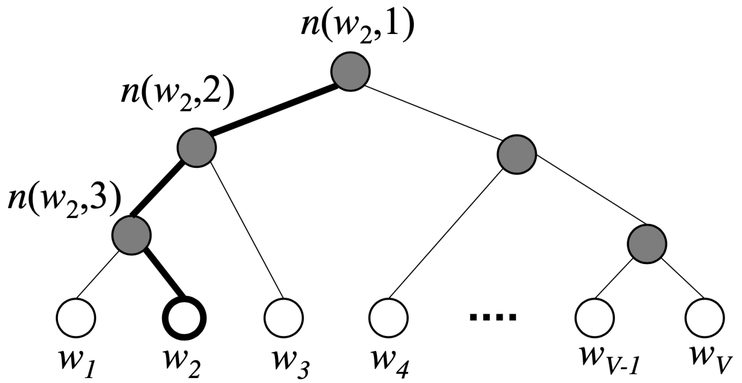


skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$
- Two algorithms:
  - Hierarchical softmax: $O(\log V)$



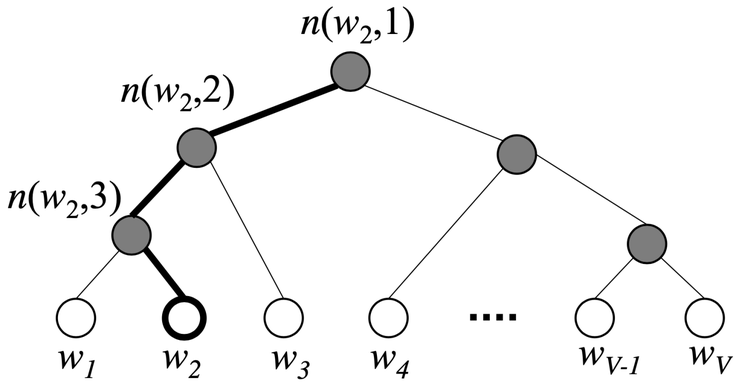
  - Negative sampling:

skip gram w/ negative sampling

$$P(w_{t+j} \mid w_t) = P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

No.

- V = 50K — 30M $O(VK)$
- Two algorithms:
  - Hierarchical softmax: $O(\log V)$



  - Negative sampling: $O(1)$

skip gram w/ negative sampling

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
    - Subsample very frequent words

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
    - Subsample very frequent words
- Randomly sample other words in the lexicon to get negative samples

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
    - Subsample very frequent words
- Randomly sample other words in the lexicon to get negative samples

(banking, regulation)

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
  - Subsample very frequent words
- Randomly sample other words in the lexicon to get negative samples

(banking, regulation)

(banking, aardvark)

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
  - Subsample very frequent words
- Randomly sample other words in the lexicon to get negative samples

(banking, regulation) (banking, aardvark)

- Binary classification rather than multiclass:

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)} \longrightarrow P(o \mid c) = \frac{1}{1 + \exp(-u_o^T v_c)} = \sigma(u_o^T v_c)$$

skip gram w/ negative sampling

- Treat the target word and a neighboring context word as positive examples.
  - Subsample very frequent words
- Randomly sample other words in the lexicon to get negative samples

(banking, regulation) (banking, aardvark)

- Binary classification rather than multiclass:

$$P(o \mid c) = \frac{\exp(u_o^T v_c)}{\sum_{i=1}^{V} \exp(u_i^T v_c)} \longrightarrow P(o \mid c) = \frac{1}{1 + \exp(-u_o^T v_c)} = \sigma(u_o^T v_c)$$

- New objective (single context word, k negative samples):

$$\log P(o_+ \mid c) + \sum_{i=1}^{k} \log(1 - P(o_i \mid c))$$

Choosing negative samples

Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*

Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

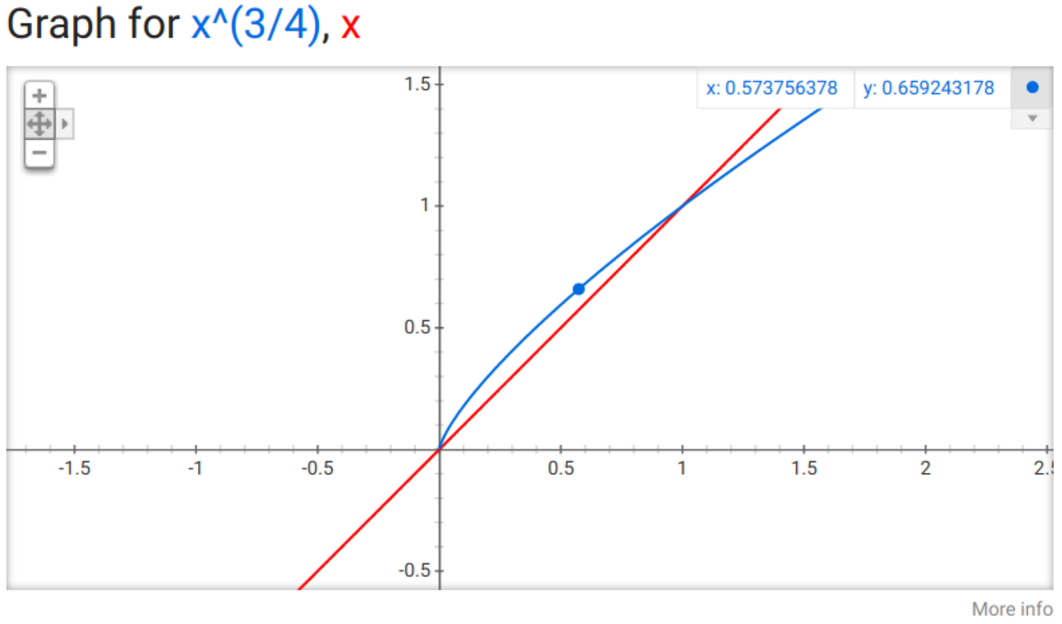
- α=0.75 works well empirically

Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

- α=0.75 works well empirically

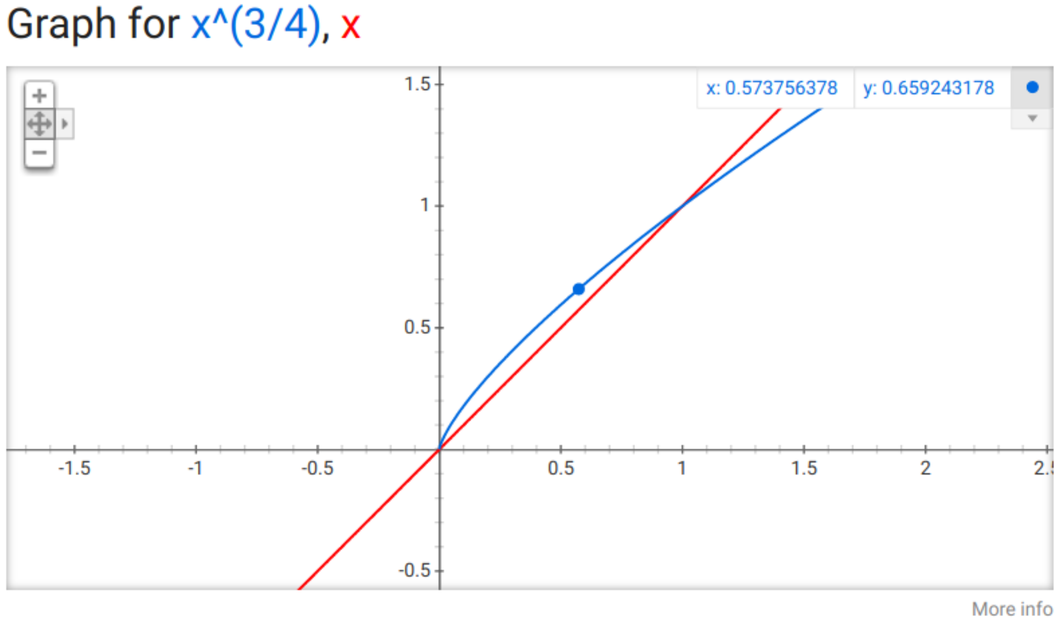


Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

- α=0.75 works well empirically
- Gives rare noise words slightly higher probability, e.g. *P(a)* = 0.99, *P(b)* = 0.01:



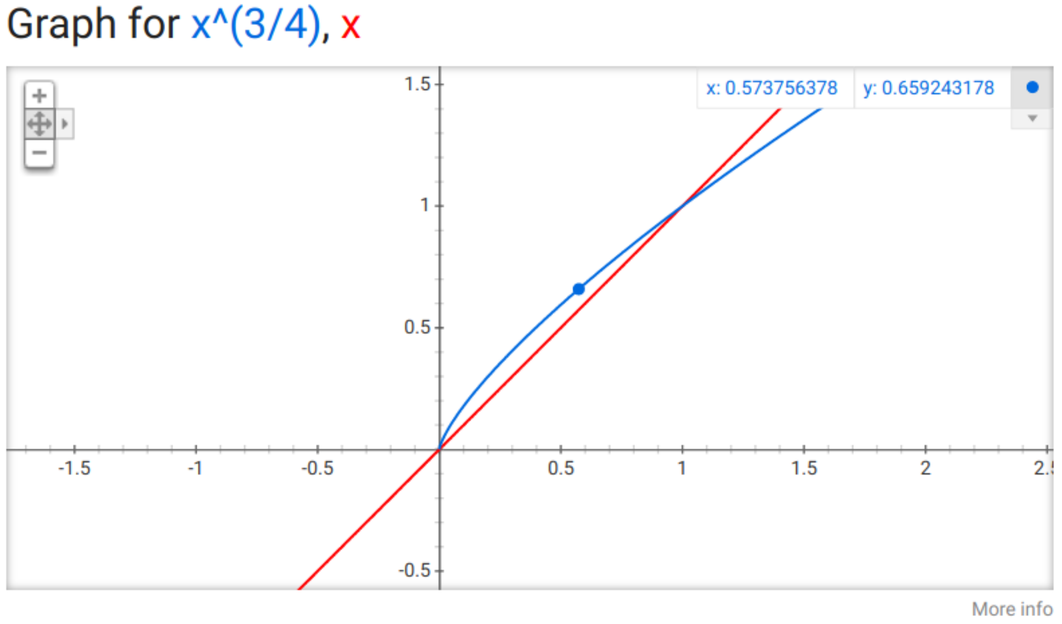
Choosing negative samples

- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

- α=0.75 works well empirically
- Gives rare noise words slightly higher probability, e.g. *P(a)* = 0.99, *P(b)* = 0.01:

$$P_\alpha(a) = \frac{0.99^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.97$$



Choosing negative samples

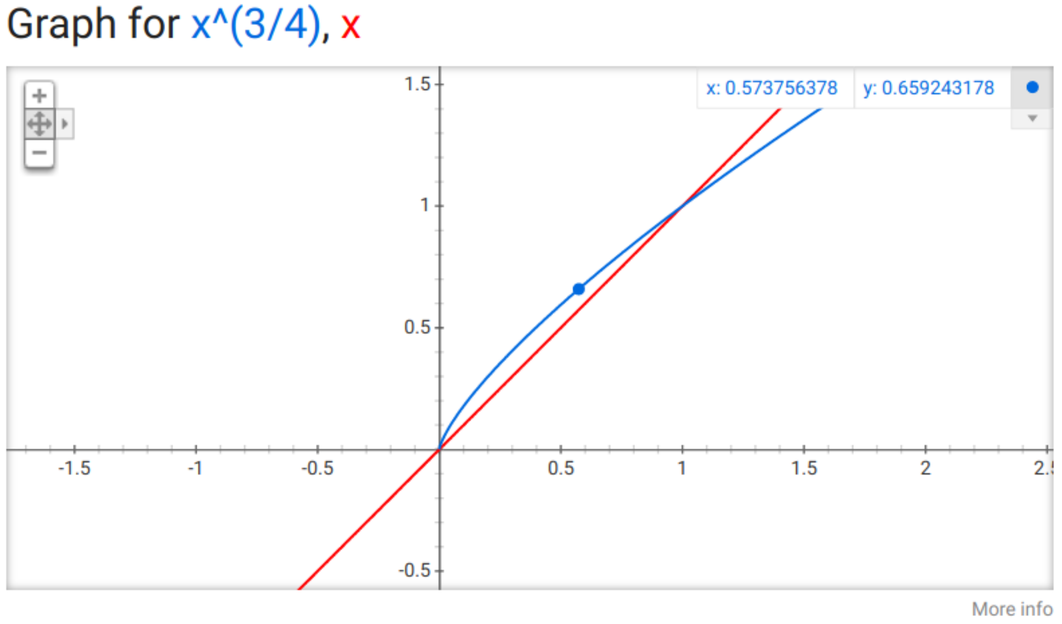
- Could pick negative samples according to unigram frequency *P(w)*
- More common to choose according to:

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

- α=0.75 works well empirically
- Gives rare noise words slightly higher probability, e.g. *P(a)* = 0.99, *P(b)* = 0.01:

$$P_\alpha(a) = \frac{0.99^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.97$$

$$P_\alpha(b) = \frac{0.01^{0.75}}{0.99^{0.75} + 0.01^{0.75}} = 0.03$$



Dense embeddings you can download!

Dense embeddings you can download!

- **word2vec** [Mikolov et al. 2013]:

  https://code.google.com/archive/p/word2vec/

Dense embeddings you can download!

- **word2vec** [Mikolov et al. 2013]:

  https://code.google.com/archive/p/word2vec/

- **GloVe** [Pennington et al. 2014]:

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

Dense embeddings you can download!

- **word2vec** [Mikolov et al. 2013]:

  https://code.google.com/archive/p/word2vec/

- **GloVe** [Pennington et al. 2014]:

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

- **fasttext** [Bojanowsi et al. 2017]:

  http://www.fasttext.cc/

Dense embeddings you can download!

- **word2vec** [Mikolov et al. 2013]:

  https://code.google.com/archive/p/word2vec/

- **GloVe** [Pennington et al. 2014]:

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

- **fasttext** [Bojanowsi et al. 2017]:

  http://www.fasttext.cc/

skiing = {^skiing$, ^ski, skii, kiin, iing, ing$}

Dense embeddings you can download!

- **word2vec** [Mikolov et al. 2013]:

  https://code.google.com/archive/p/word2vec/

- **GloVe** [Pennington et al. 2014]:

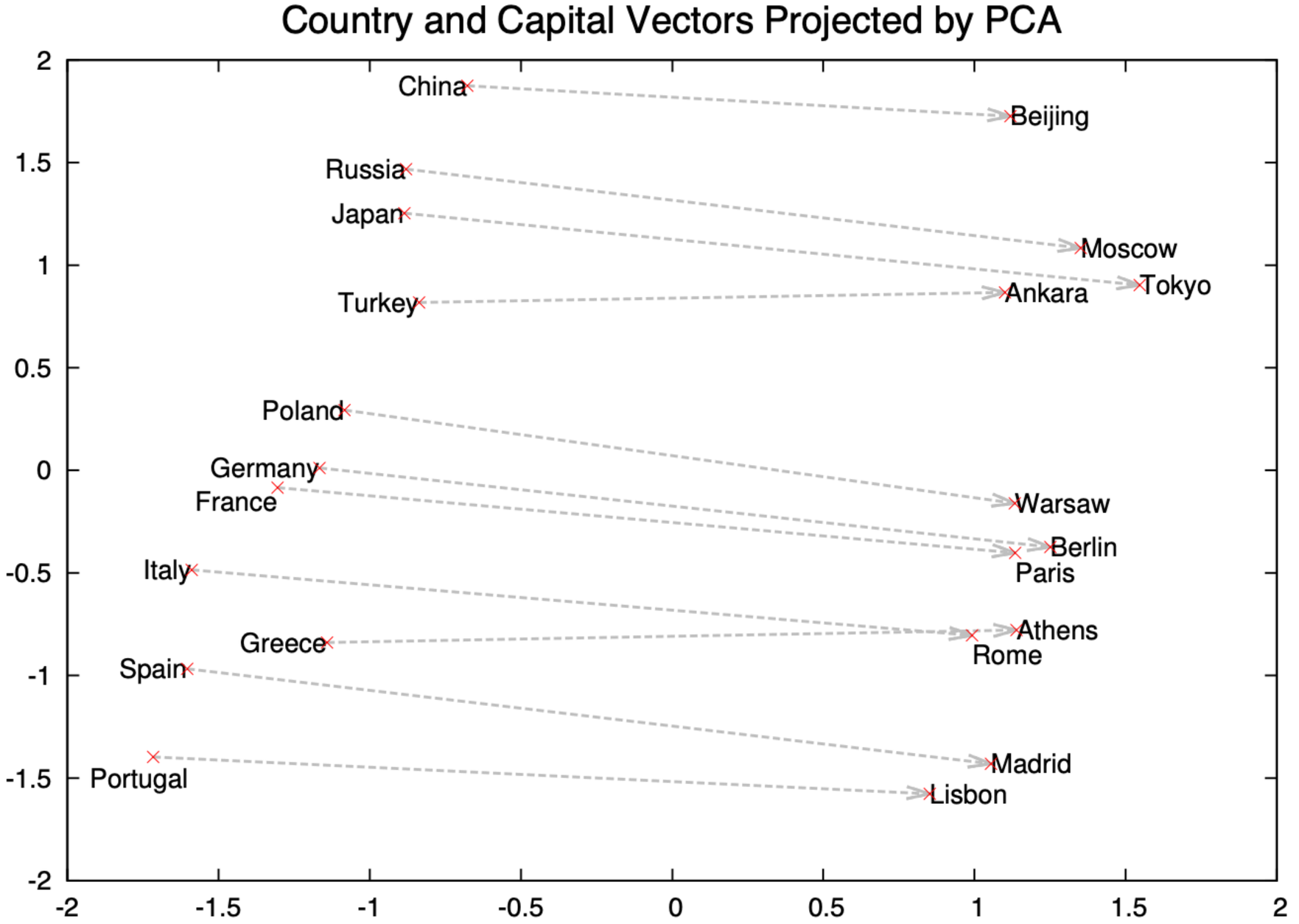
  http://nlp.stanford.edu/projects/glove/

- **fasttext** [Bojanowsi et al. 2017]:

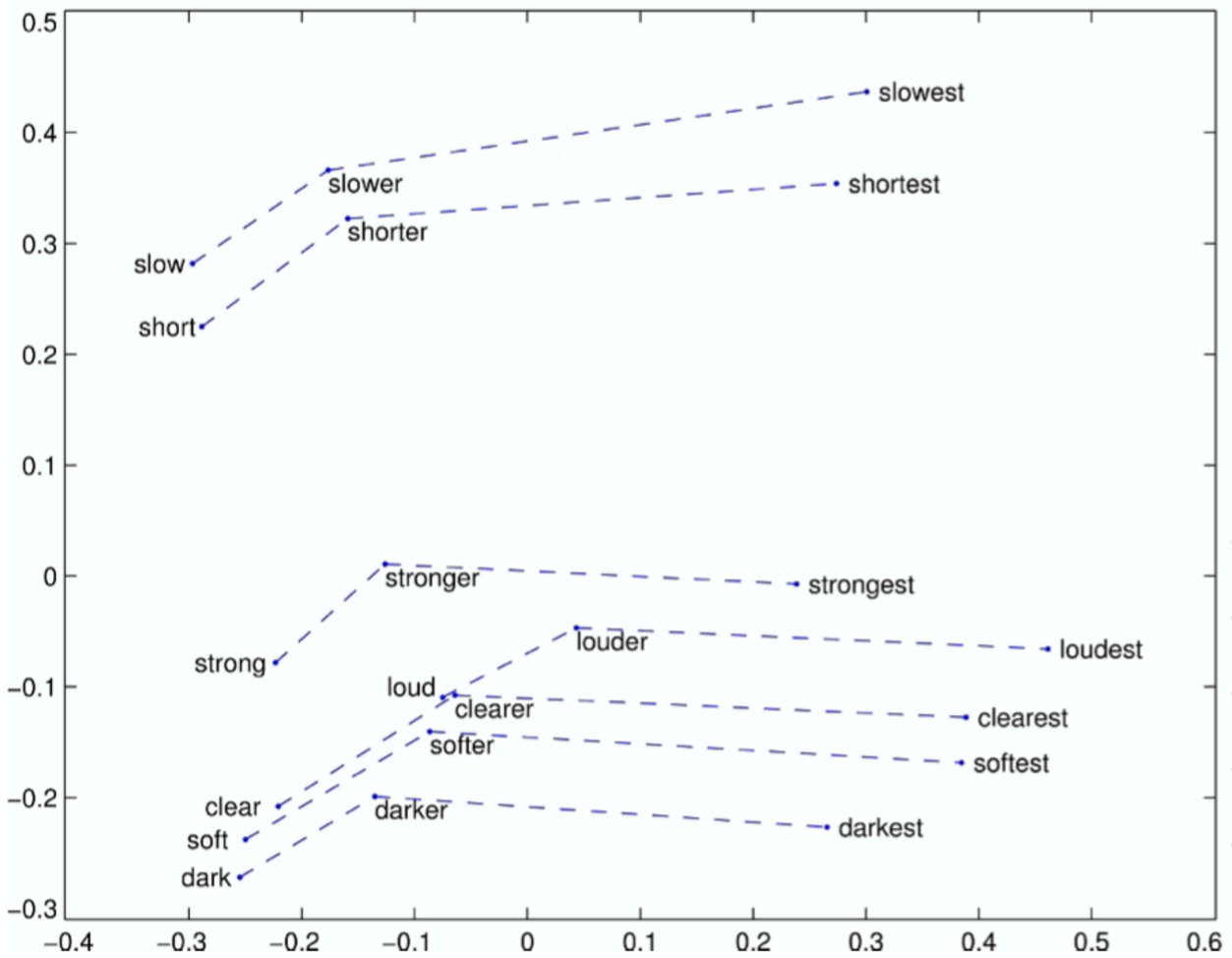
  http://www.fasttext.cc/

	Singular+neut	Plural+neut	
Nominative	предложение	предложения	sentence (s)
Genitive	предложения	предложений	(of) sentence (s)
Dative	предложению	предложениям	(to) sentence (s)
Accusative	предложение	предложения	sentence (s)
Instrumental	предложением	предложениями	(by) sentence (s)
Prepositional	предложении	предложениях	(in/at) sentence (s)

skiing = {^skiing$, ^ski, skii, kiin, iing, ing$}



[Mikolov et al. 2013]



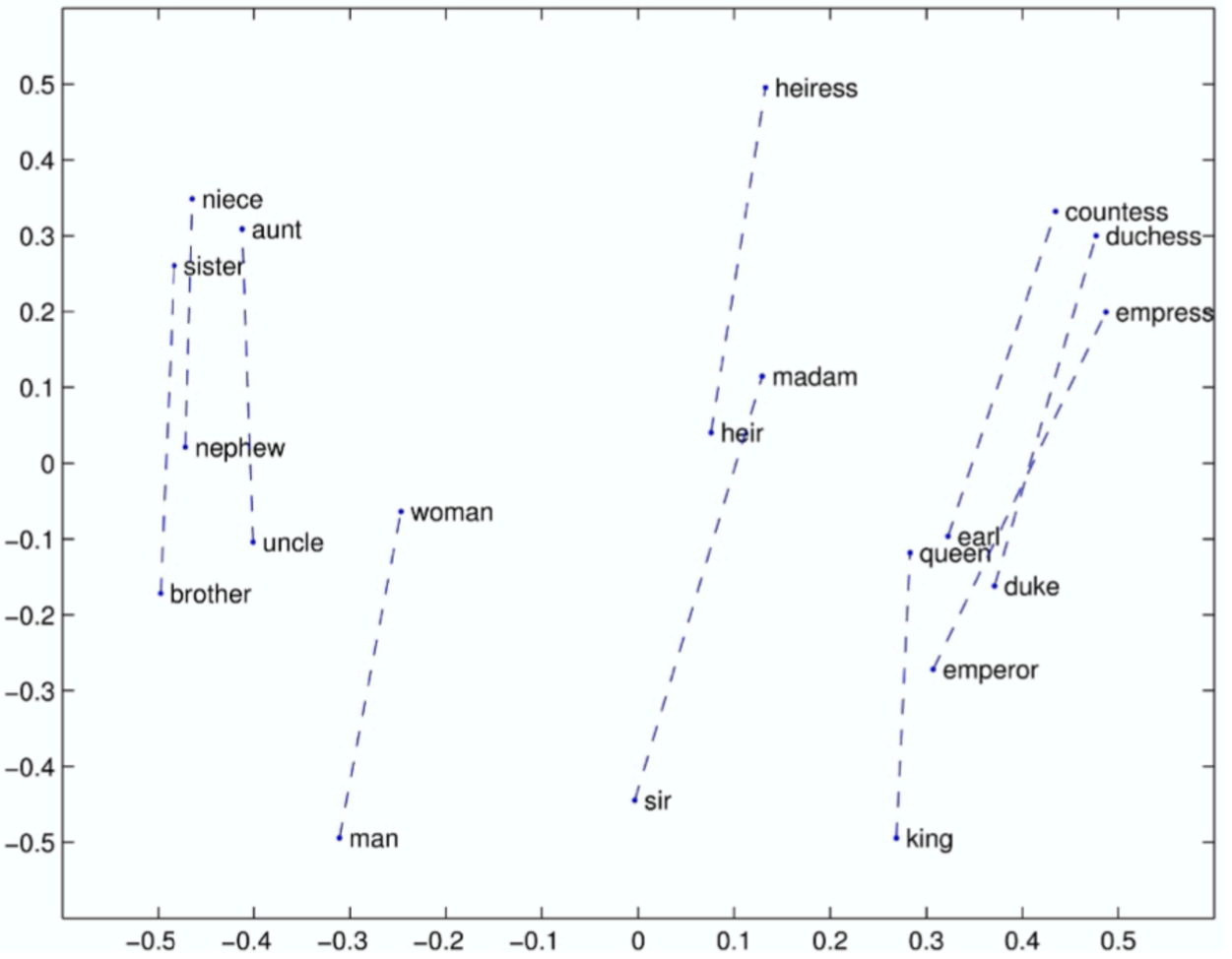
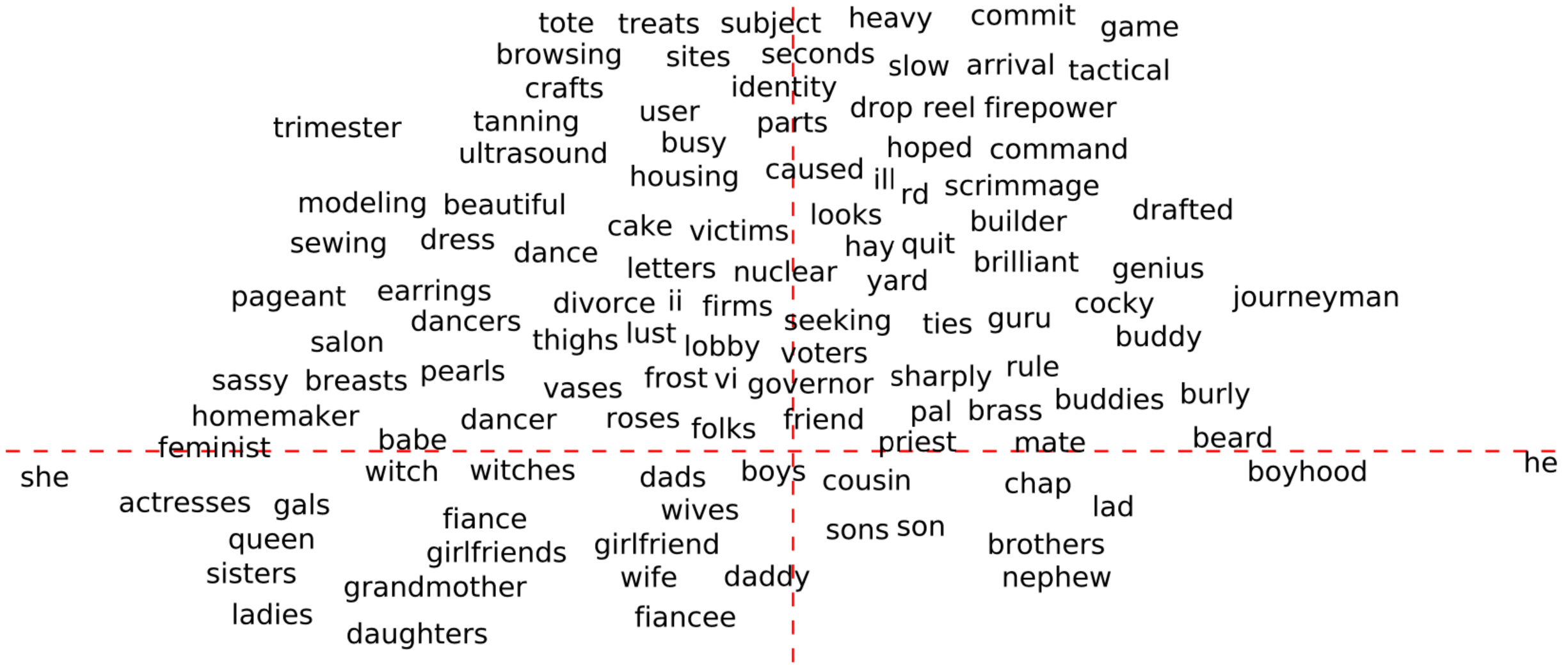


Image: http://ruder.io/secret-word2vec



$$\overrightarrow{computer\ programmer} - \overrightarrow{man} + \overrightarrow{woman} = \overrightarrow{homemaker}$$

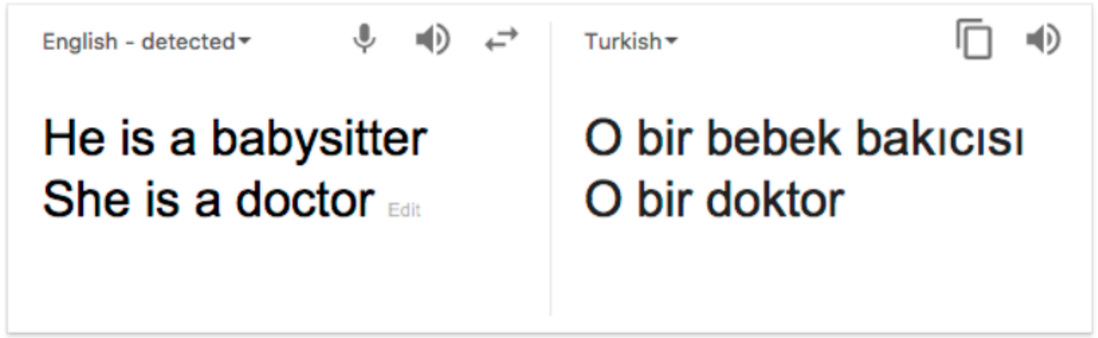
[Bolukbasi et al. 2016]

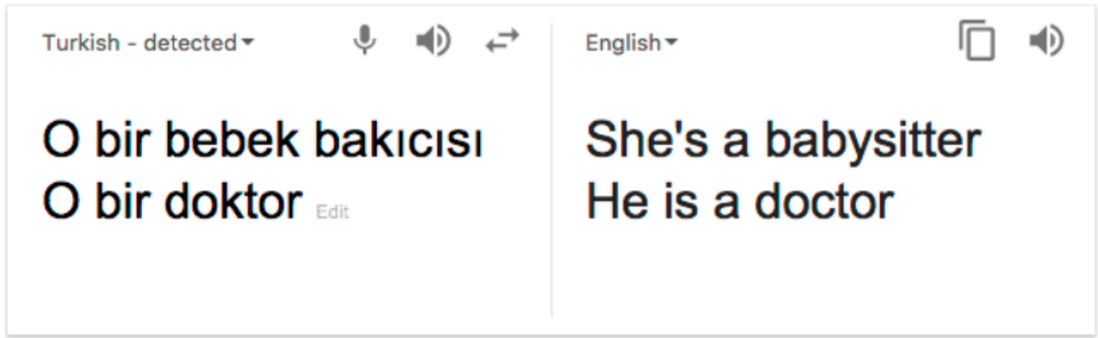
Bias in word embeddings

Malay example credit: Jecelyn Yeen https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

Turkish example credit: Neelima Jadhav https://www.slideshare.net/NeelimaJadhav1/bias-in-artificial-intelligence

Bias in word embeddings

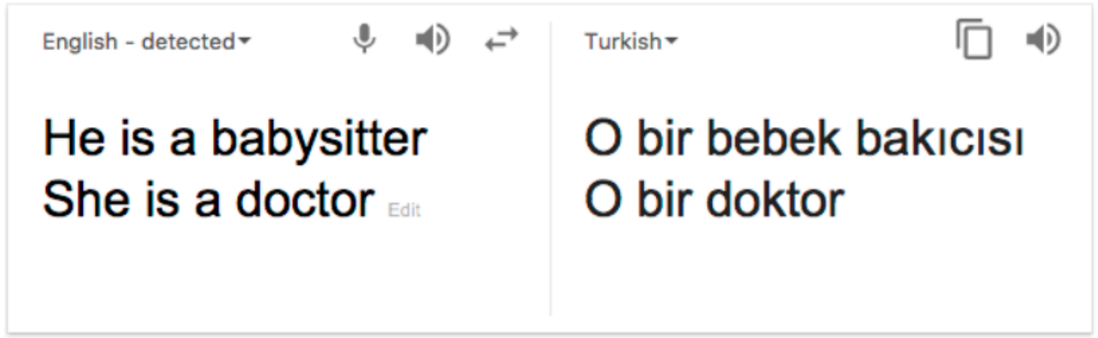


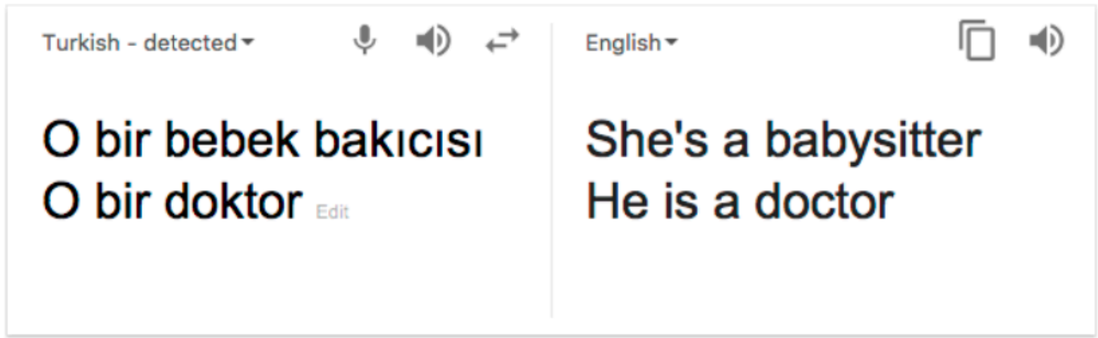


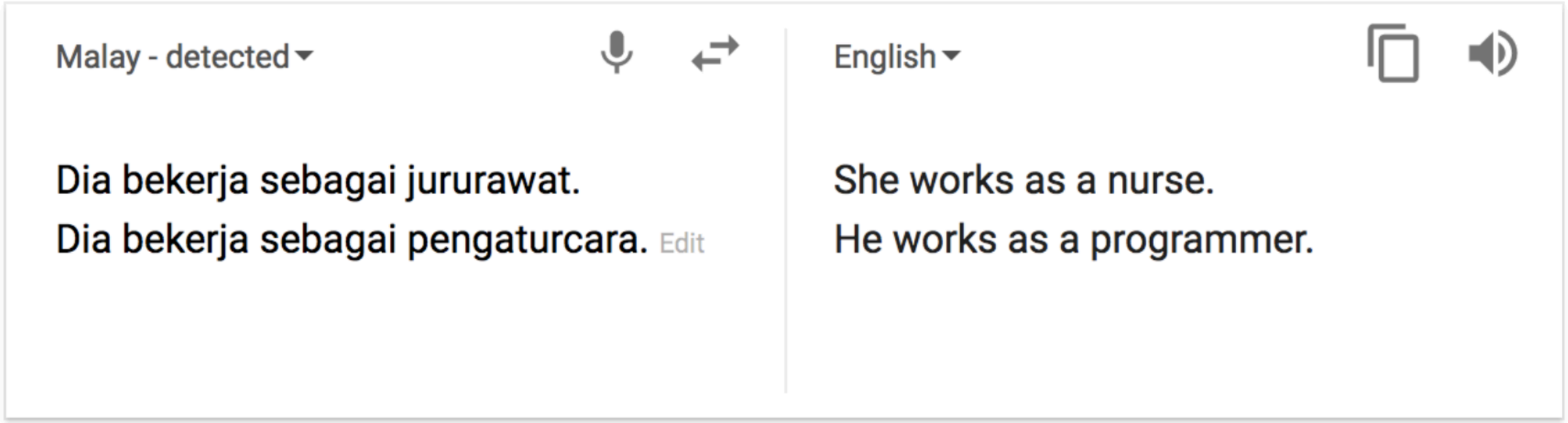
Malay example credit: Jecelyn Yeen https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

Turkish example credit: Neelima Jadhav https://www.slideshare.net/NeelimaJadhav1/bias-in-artificial-intelligence

Bias in word embeddings



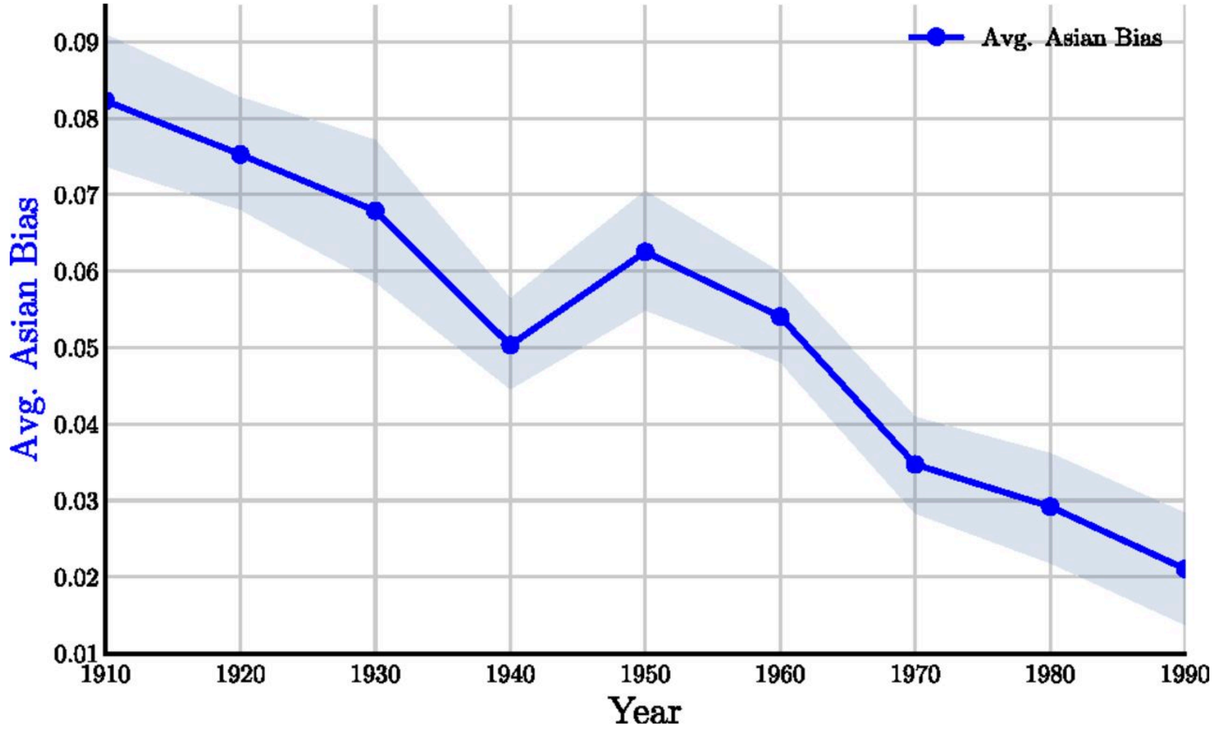




Malay example credit: Jecelyn Yeen https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c

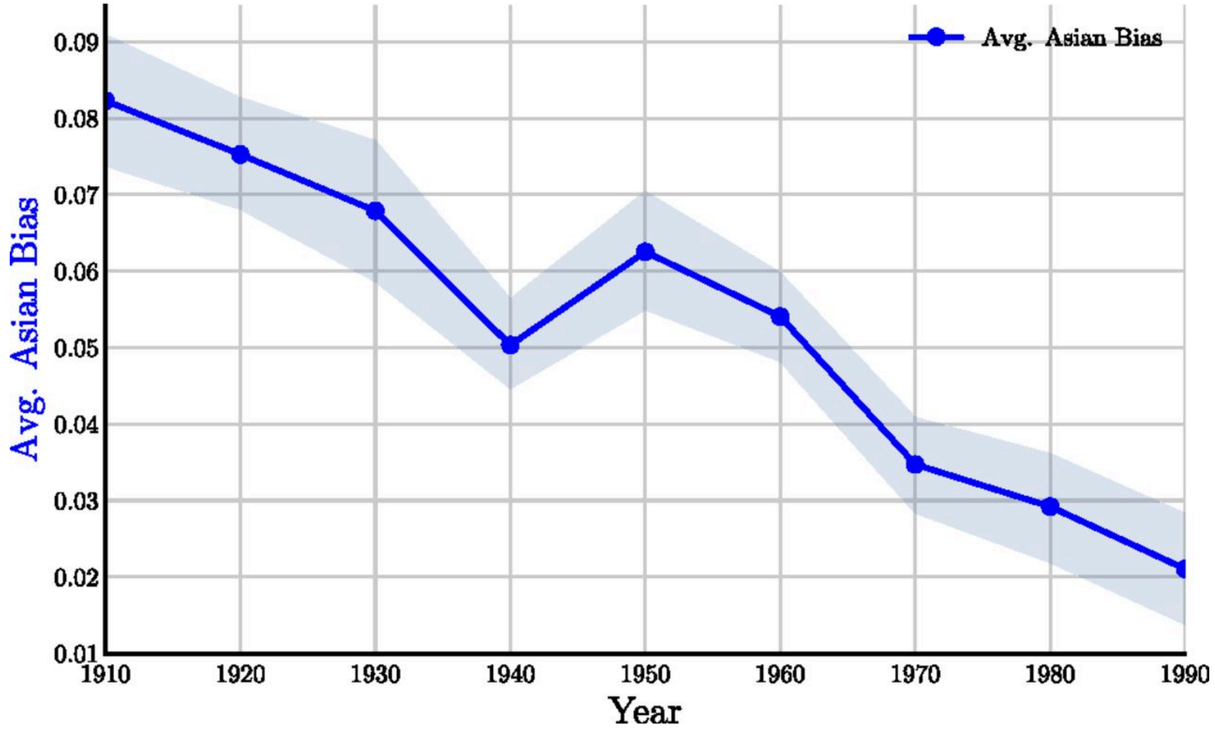
Turkish example credit: Neelima Jadhav https://www.slideshare.net/NeelimaJadhav1/bias-in-artificial-intelligence

Change in linguistic framing 1910-1990



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

Change in linguistic framing 1910-1990



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

ML algorithms shown to **amplify** bias in data! [Zhao et al. 2017]

Evaluating word vectors

Evaluating word vectors

- **Intrinsic evaluations** test whether the representations align with our intuitions about word meaning.

Evaluating word vectors

- **Intrinsic evaluations** test whether the representations align with our intuitions about word meaning.
  - How well does cosine similarity of word embeddings correlate with human judgements?

Evaluating word vectors

- **Intrinsic evaluations** test whether the representations align with our intuitions about word meaning.
  - How well does cosine similarity of word embeddings correlate with human judgements?
  - Completing analogies: a : b :: c : ?

Evaluating word vectors

- **Intrinsic evaluations** test whether the representations align with our intuitions about word meaning.
  - How well does cosine similarity of word embeddings correlate with human judgements?
  - Completing analogies: a : b :: c : ?
- **Extrinsic evaluations** test whether the representations are useful for downstream tasks, such as tagging, parsing, coreference, question answering, …

Evaluating word vectors

- **Intrinsic evaluations** test whether the representations align with our intuitions about word meaning.
  - How well does cosine similarity of word embeddings correlate with human judgements?
  - Completing analogies: a : b :: c : ?
- **Extrinsic evaluations** test whether the representations are useful for downstream tasks, such as tagging, parsing, coreference, question answering, …
  - Provide embeddings W_1, W_2 as input to the same classifier (retrained), how well does each perform?

Announcements

Announcements

- Don’t forget we will hold recitations Friday to discuss P1.

Announcements

- Don’t forget we will hold recitations Friday to discuss P1.
- Working on getting lectures posted on YouTube (CMU-only) w/ closed captions.