Lexical and Vector Semantics

CSE538 - Spring 2025
Natural Language Processing

Topics

- Lexical Ambiguity (why word sense disambiguation)
- Word Vectors
- Topic Modeling

Objectives

- Define common semantic tasks in NLP and learn some approaches to solve.
- Understand linguistic information necessary for semantic processing
- Motivate deep learning models necessary to capture language semantics.
- Learn word embeddings (the starting point for modern large language models)

Terminology: lemma and wordform

- A **lemma** or **citation form**
  - Same stem, part of speech, rough semantics
- A **wordform**
  - The inflected word as it appears in text

Wordform	Lemma
banks	bank
sung	sing
duermes	dormir

(Jurafsky & Martin, SLP, 2019)

Lemmas have senses

- One lemma "bank" can have many meanings:

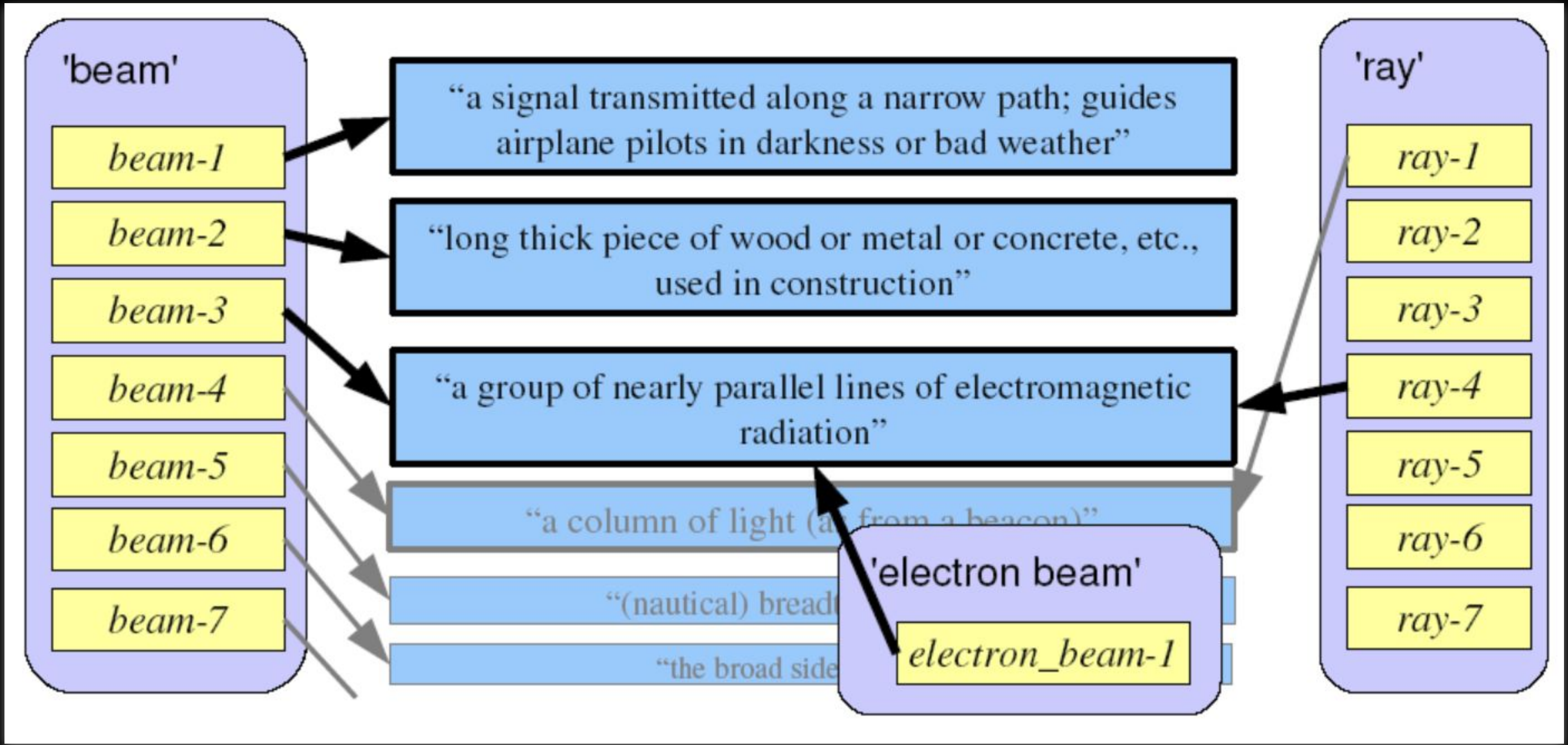
Sense 1:
  - …a **bank**[1] can hold the investments in a custodial account…

Sense 2:
  - "…as agriculture burgeons on the east **bank**[2] the river will shrink even more"

- **Sense** (or **word sense**)
  - A discrete representation of an aspect of a word's meaning.
- The lemma **bank** here has two senses

(Jurafsky & Martin, SLP, 2019)

Lemmas have senses



(Schwartz, 2011)

Homonymy

Homonyms: words that share a form but have unrelated, distinct meanings:

- $bank_1$: financial institution, $bank_2$: sloping land
- bat_1: club for hitting a ball, bat_2: nocturnal flying mammal

1. Homographs (bank/bank, bat/bat)
2. Homophones:
   1. Write and right
   2. Piece and peace

(Jurafsky & Martin, SLP, 2019)

Homonymy causes problems for NLP applications

- Information retrieval
  - "`bat care`"
- Machine Translation
  - `bat`: murciélago (animal) or bate (for baseball)
- Text-to-Speech
  - `bass` (stringed instrument) vs. `bass` (fish)

(Jurafsky & Martin, SLP, 2019)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

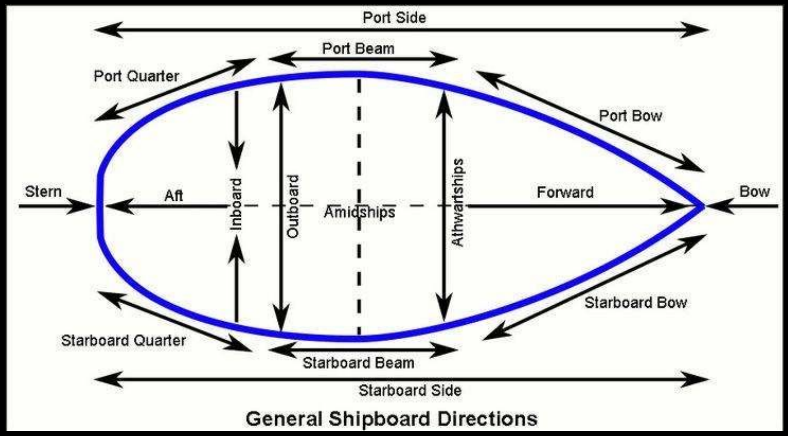
He walked along the **port** next to the steamer.

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

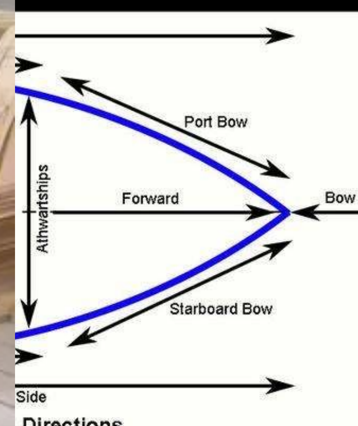


Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.



Word Sense Disambiguation

He put the **port** on the ship.
He walked along the **port** of the steamer.
He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

Word Sense Disambiguation

He put the **port** on the ship.

He walked along the **port** of the steamer.

He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

Word Sense Disambiguation

He put the **port** on the ship.
He walked along the **port** of the steamer.
He walked along the **port** next to the steamer.

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

Word Sense Disambiguation

He put the **port** on the ship.
He walked along the **port** of the steamer.
He walked along the **port** next to the steamer.

As a verb…

1. **port** (put or turn on the left side, of a ship) *"port the helm"*
2. **port** (bring to port) *"the captain ported the ship at night"*
3. **port** (land at or reach a port) *"The ship finally ported"*
4. **port** (turn or go to the port or left side, of a ship) *"The big ship was slowly porting"*
5. **port** (carry, bear, convey, or bring) *"The small canoe could be ported easily"*
6. **port** (carry or hold with both hands diagonally across the body, especially of weapons) *"port a rifle"*
7. **port** (drink port) *"We were porting all in the club after dinner"*
8. **port** (modify (software) for use on a different machine or platform)

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

Objective

great

great.a.1 (relatively large in size or number or extent; larger than others of its kind)

great.a.2, outstanding (of major significance or importance)

great.a.3 (remarkable or out of the ordinary in degree or magnitude or effect)

bang-up, bully, corking, cracking, dandy, **great.a.4**, groovy, keen, neat, nifty, not bad, peachy, slap-up, swell, smashing, old (very good)

capital, **great.a.5**, majuscule (uppercase)

big, enceinte, expectant, gravid, **great.a.6**, large, heavy, with child (in an advanced stage of pregnancy)

Objective

great

great.a.1 (relatively large in size or number or extent; larger than others of its kind)

great.a.2, outstanding (of major significance or importance)

great.a.3 (remarkable or out of the ordinary in degree or magnitude or effect)

bang-up, bully, corking, cracking, dandy, **great.a.4**, groovy, keen, neat, nifty, not bad, peachy, slap-up, swell, smashing, old (very good)

capital, **great.a.5**, majuscule (uppercase)

big, enceinte, expectant, gravid, **great.a.6**, large, heavy, with child (in an advanced stage of pregnancy)

great.n.1 (a person who has achieved distinction and honor in some field) port.n.1

port.n.2

port.n.3,

port.n.4

port.n.5

Word Sense Disambiguation

A classification problem:

General Form:

f(sent_tokens, (target_index, lemma, POS)) -> word_sense

He walked along the **port** next to the steamer.

Word Sense Disambiguation

A classification problem:

General Form:

f(sent_tokens, (target_index, lemma, POS)) -> word_sense

Logistic Regression (or any discriminative classifier):

$P_{lemma,POS}(\text{sense} = s \mid \text{features})$

He walked along the **port** next to the steamer.

Word Sense Disambiguation

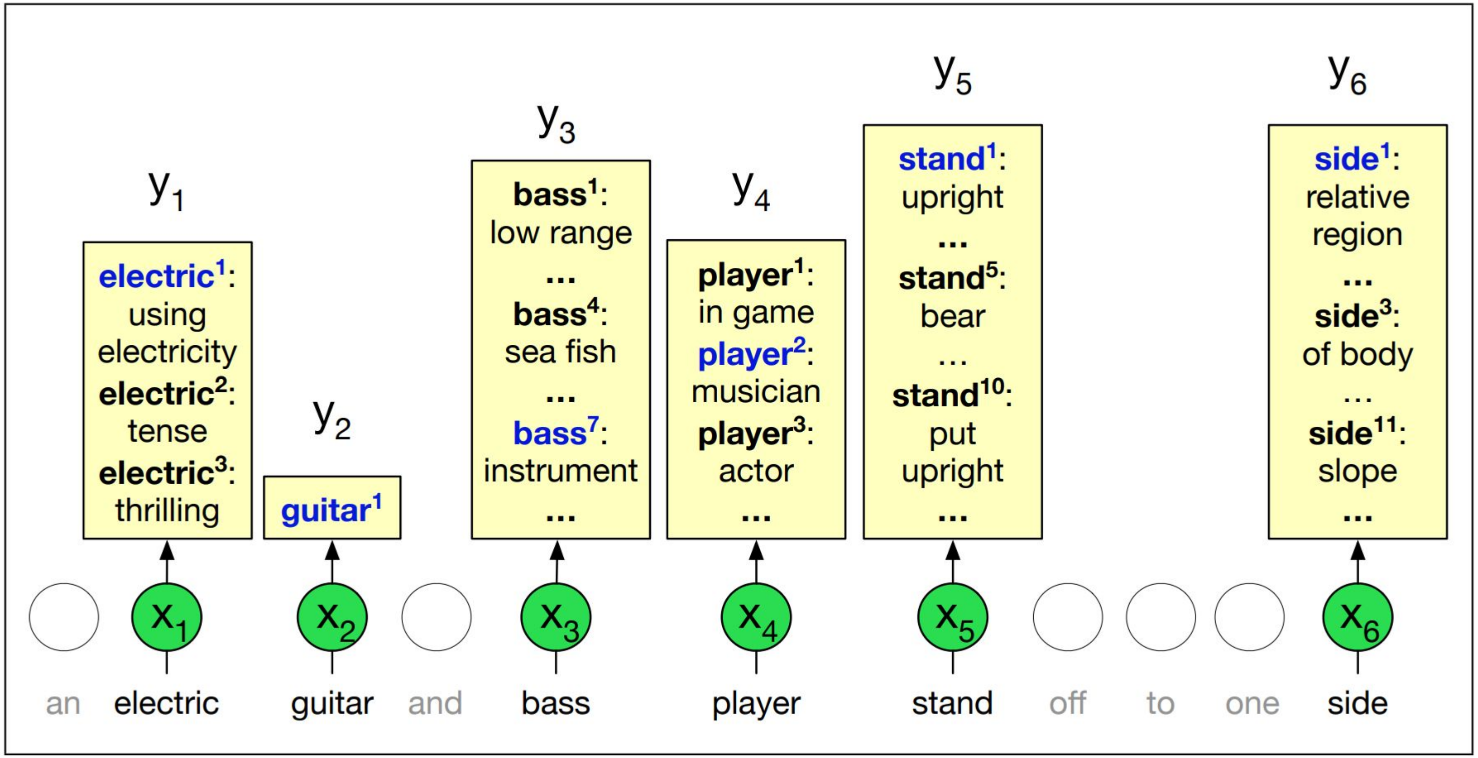


Figure 19.8 The all-words WSD task, mapping from input words (*x*) to WordNet senses (*y*). Only nouns, verbs, adjectives, and adverbs are mapped, and note that some words (like *guitar* in the example) only have one sense in WordNet. Figure inspired by Chaplot and Salakhutdinov (2018).

(Jurafsky, SLP 3)

Distributional Hypothesis:

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Distributional Hypothesis:

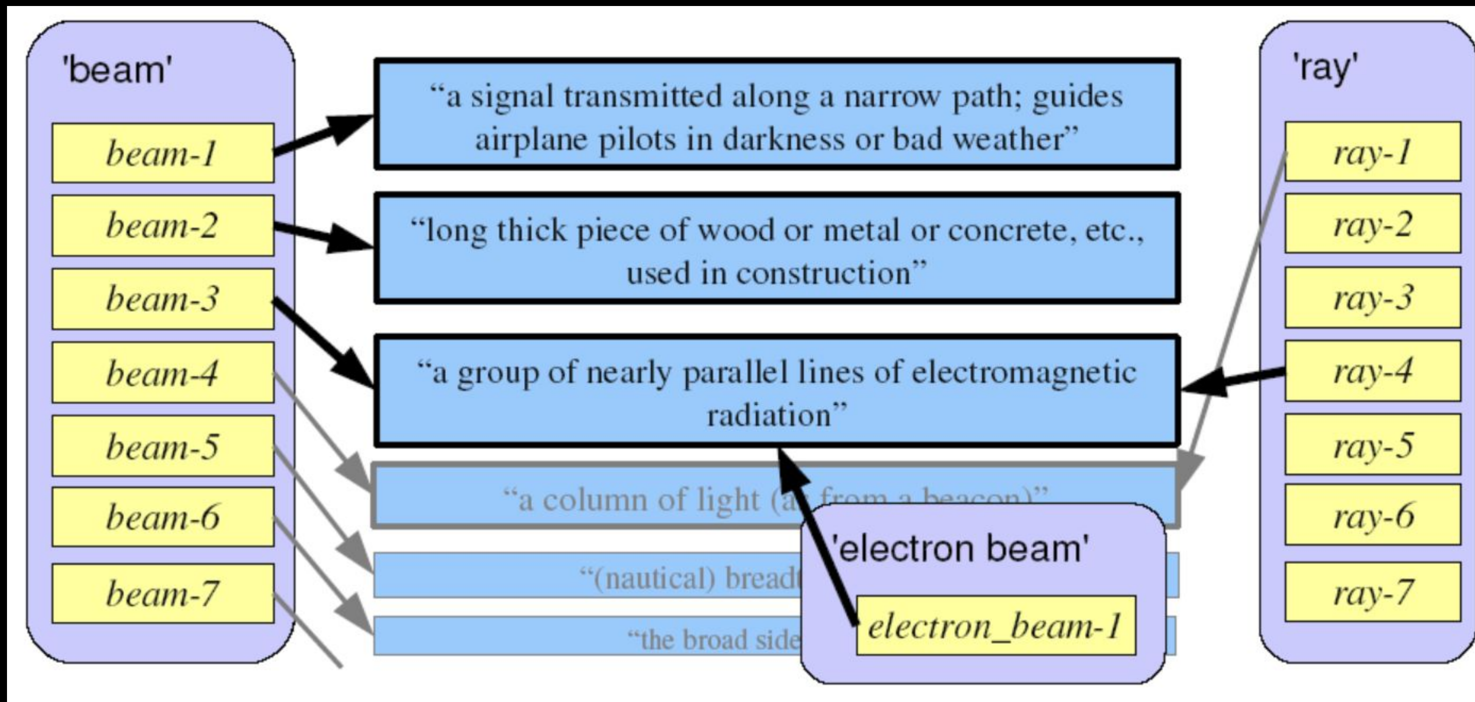
Wittgenstein, 1945: “*The meaning of a word is its use in the language*”

Distributional hypothesis -- A word’s meaning is defined by all the different contexts it appears in (i.e. how it is “distributed” in natural language).

Firth, 1957: “*You shall know a word by the company it keeps*”

The nail hit the beam behind the wall.

Distributional Hypothesis



The nail hit the beam behind the wall.

Distributional Hypothesis

Similarity -

Relatedness -

The nail hit the beam behind the wall.

Distributional Hypothesis

Similarity - Has same or similar meaning.
synonyms (*same as*), hypernyms (*is-a*), hyponyms (*has-a*)

Relatedness - Any relationship:
includes similarity but also antonyms, meronyms (*part-of*), etc….

The nail hit the beam behind the wall.

Distributional Hypothesis

Similarity - Has same or similar meaning.

synonyms (*same as*), hypernyms (*is-a*), hyponyms (*has-a*)

beam is-a *piece of wood*

beam is similar to *piece of wood*

Relatedness - Any relationship:

includes similarity but also antonyms, meronyms (*part-of*), etc….

The nail hit the beam behind the wall.

Distributional Hypothesis

Similarity - Has same or similar meaning.

synonyms (*same as*), hypernyms (*is-a*), hyponyms (*has-a*)

beam is-a *piece of wood*

beam is similar to *piece of wood*

Relatedness - Any relationship:

includes similarity but also antonyms, meronyms (*part-of*), etc….

beam is-part-of *a house*

beam is related to *a house*

~~*beam* is similar to *a house*~~

The nail hit the beam behind the wall.

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. Lesk algorithm
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. **Lesk algorithm**
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Lesk Algorithm for WSD

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence

  return(best-sense)
```

Figure 19.10 The Simplified Lesk algorithm. The COMPUTEOVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the *context* in a more complex way.

Lesk Algorithm for WSD

function SIMPLIFIED LESK(*word, sentence*) **returns** best sense of *word*

best-sense ← most frequent sense for *word*
max-overlap ← 0
context ← set of words in *sentence*
for each *sense* **in** senses of *word* **do**
  signature ← set of words in the gloss and examples of *sense*

return(*best-sense*)

Figure 19.10 The Simplified Lesk algorithm. The COMPUTEOVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the *context* in a more complex way.

Lesk Algorithm for WSD

```
function SIMPLIFIED LESK(word, sentence) returns best sense of word

  best-sense ← most frequent sense for word
  max-overlap ← 0
  context ← set of words in sentence
  for each sense in senses of word do
    signature ← set of words in the gloss and examples of sense
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
          max-overlap ← overlap
          best-sense ← sense
  end
  return(best-sense)
```

Figure 19.10 The Simplified Lesk algorithm. The COMPUTEOVERLAP function returns the number of words in common between two sets, ignoring function words or other words on a stop list. The original Lesk algorithm defines the *context* in a more complex way.

Lesk Algorithm for WSD

- bank.n.1 (sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
- bank.n.2 (a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"

```
    overlap ← COMPUTEOVERLAP(signature, context)
    if overlap > max-overlap then
        max-overlap ← overlap
        best-sense ← sense
  end
  return(best-sense)
```

The bank can guarantee deposits will cover future tuition costs, ...

- bank.n.1 (sloping land (especially the slope beside a body of water)) "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"
- bank.n.2 (a financial institution that accepts deposits and channels the money into lending activities) "he cashed a check at the bank"; "that bank holds the mortgage on my home"
- ...
- bank.n.4 (an arrangement of similar objects in a row or in tiers) "he operated a bank of switches"
- ...
- bank.n.8 (a building in which the business of banking transacted) "the bank is on the corner of Nassau and Witherspoon"
- bank.n.9 (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) "the plane went into a steep bank"

```
end
return(best-sense)
```

The bank can guarantee deposits will cover future tuition costs, ...

Lesk Algorithm for WSD

- **striker.n.1** (a forward on a soccer team)
- **striker.n.2** (someone receiving intensive training for a naval technical rating)
- **striker.n.3** (an employee on strike against an employer)
- **striker.n.4** (someone who hits) *"a hard hitter"; "a fine striker of the ball"; "blacksmiths are good hitters"*
- **striker.n.5** (the part of a mechanical device that strikes something)

$\quad$ *overlap* ← ComputeOverlap(*signature*, *context*)
$\quad$ **if** *overlap* > *max-overlap* **then**
$\quad\quad$ *max-overlap* ← *overlap*
$\quad\quad$ *best-sense* ← *sense*
end
return(*best-sense*)

He addressed the strikers at the rally.

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. **Lesk algorithm**
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Approaches to WSD

I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. Lesk algorithm
   E.g. compare context to sense definitions.
4. **Selectors -- other *target words* that appear with same context** *E.g. counts for any selector.*
5. Contextual Embeddings
   E.g. real valued vectors that "encode" the context (TBD).

Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse

Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse

object of

He addressed the strikers at the rally.

Selectors

… a word which can take the place of another given word within the same local context (Lin, 1997)

Original version: Local context defined by dependency parse (Lin, 1997)

Web version: Local context defined by lexical patterns matched on the Web (Schwartz, 2008).

*"He addressed the * at the rally."*

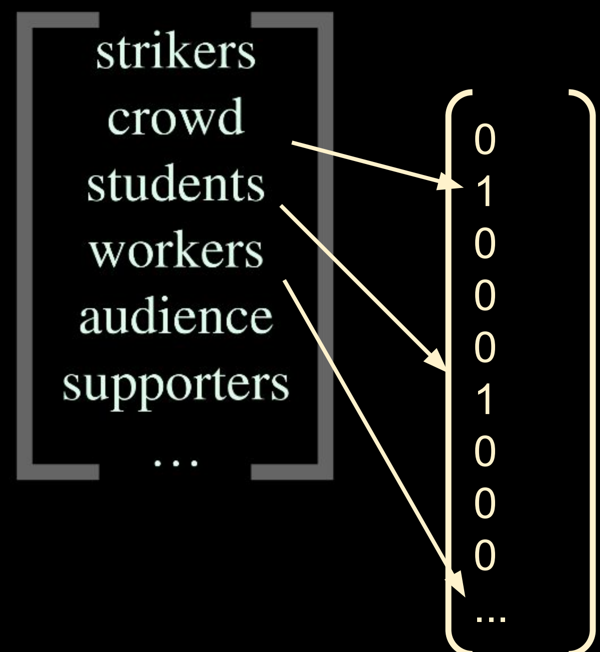
Selectors

"He addressed the strikers at the rally."

$$\begin{bmatrix} \text{strikers} \\ \text{crowd} \\ \text{students} \\ \text{workers} \\ \text{audience} \\ \text{supporters} \\ \ldots \end{bmatrix}$$

Selectors

"He addressed the strikers at the rally."



Selectors

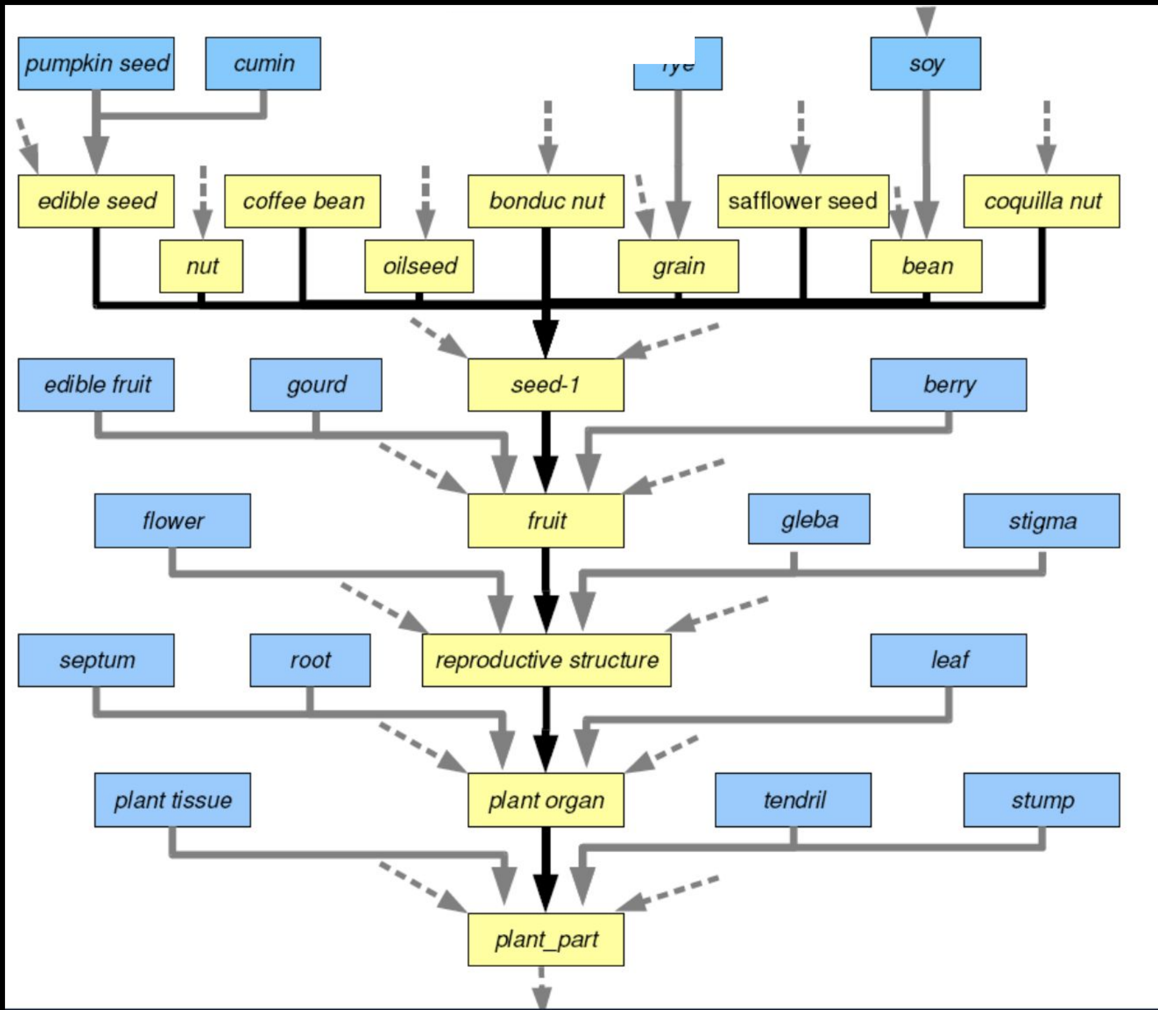
"He addressed the strikers at the rally."

he	addressed	strikers	rally
man	scolded	crowd	protest
owners	rallyied	students	demonstration
Mary	kept	workers	work
...	...	audience	stadium
		supporters	...
		...	

Selectors

Leverages *hypernymy:*

concept1 <is-a> concept2



Why Are Selectors Effective?

Sets of selectors tend to vary extensively by word sense:

bill-n.1	*bill-n.2*	*bill-n.3*
bill	bill	**market**
it	**staff**	system
legislation	system	**paper**
system	**money**	**note**
program	**time**	bill
law	it	**bond**
plan	**tax**	**stock**
you	**work**	**debt**
measure	**rent**	**rate**
project	**tuition**	**report**

occur-v.1	*occur-v.2*	*occur-v.3*
be	go	go
happen	get	**look**
occur	Come	**break**
go	have	**remove**
take	**try**	**find**
work	**lead**	get
come	**listen**	**place**
see	work	**keep**
have	be	**stick**
change	**belong**	**stop**

- *Polls show wide, generalized support for some vague concept of service, but the **bill** now under discussion lacks any passionate public backing.*
  training set never contained: “but the _ now under”

- *… in his lecture, refers to the “startling experience which almost every person confesses, that particular passages of conversation and action have **occurred** to him in the same order before, whether dreaming or waking …*
  small context is contradictory:
    “action have occurred” => occur-v.1 (“to happen or take place”)
    “occurred to him” => occur-v.2 (“to come to mind”)

bill-n.1	***bill-n.2***	***bill-n.3***
bill	bill	**market**
it	**staff**	system
legislation	system	**paper**
system	**money**	**note**
program	**time**	bill
law	it	**bond**
plan	**tax**	**stock**
you	**work**	**debt**
measure	**rent**	**rate**
project	**tuition**	**report**

occur-v.1	***occur-v.2***	***occur-v.3***
be	go	go
happen	get	**look**
occur	Come	**break**
go	have	**remove**
take	**try**	**find**
work	**lead**	get
come	**listen**	**place**
see	work	**keep**
have	be	**stick**
change	**belong**	**stop**

Vector Semantics

1. Word2vec
2. Topic Modeling - Latent Dirichlet Allocation (LDA)

Supervised Selectors

	base	w/ sels	*mfs*	*tests*
noun	87.9	**91.7**	*80.9*	*2559*
verb	83.3	**83.7**	*76.5*	*2292*
both	85.7	**87.9**	*78.8*	*4851*

Accuracy over SemEval-2007: Task 17.

Approaches to WSD

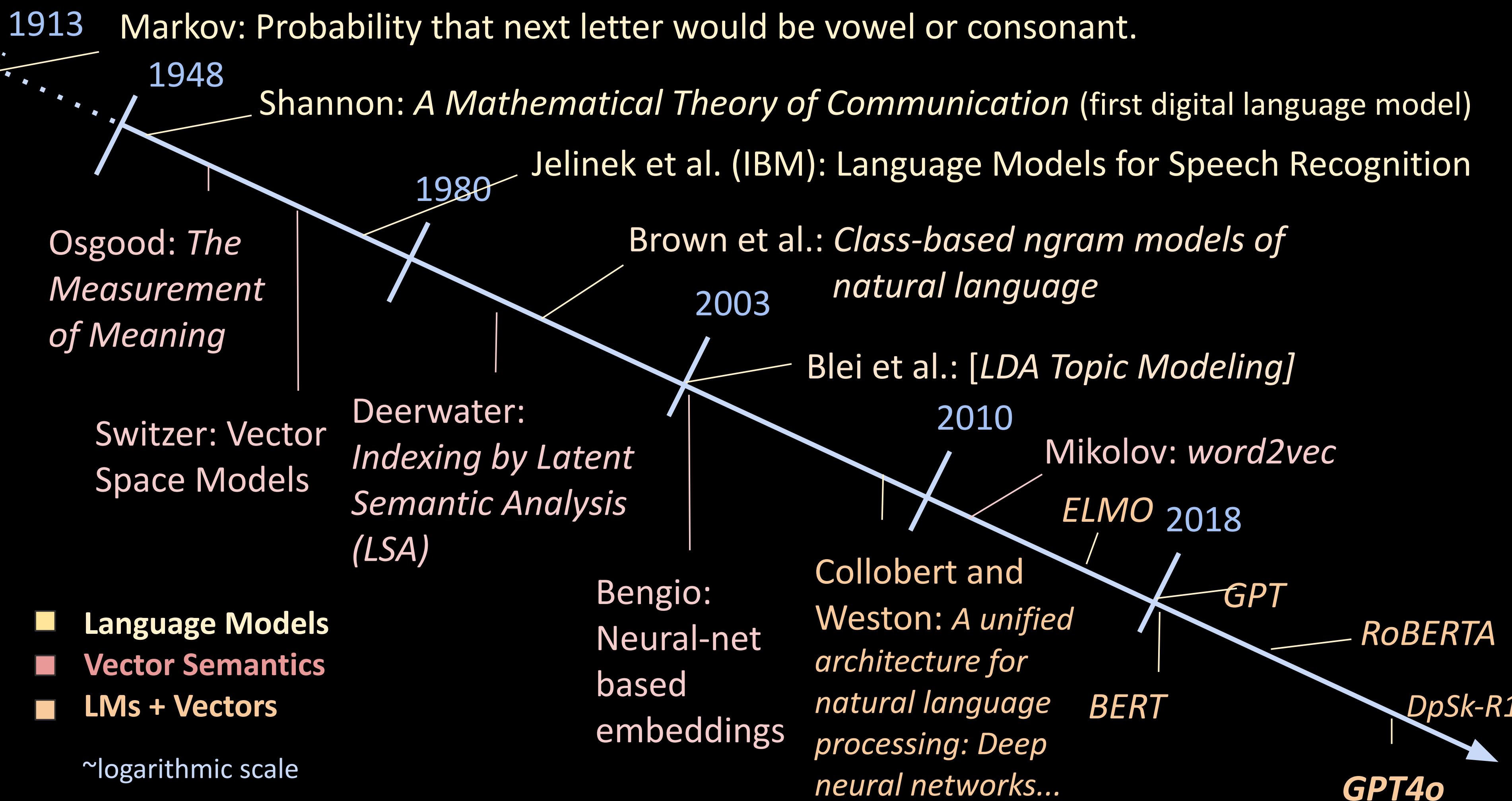
I.e. how to operationalize the distributional hypothesis.

1. Bag of words for context
   E.g. multi-hot for any word in a defined "context".
2. Surrounding window with positions
   E.g. one-hot per position relative to word).
3. Lesk algorithm
   E.g. compare context to sense definitions.
4. Selectors -- other *target words* that appear with same context
   E.g. counts for any selector.
5. Contextual Embeddings - *introduced with Transformer LMs*
   E.g. real valued vectors that "encode" the context (TBD).

Vector Semantics

1. Word2vec
2. Topic Modeling - Latent Dirichlet Allocation (LDA)

Timeline: *Language Modeling* and *Vector Semantics*



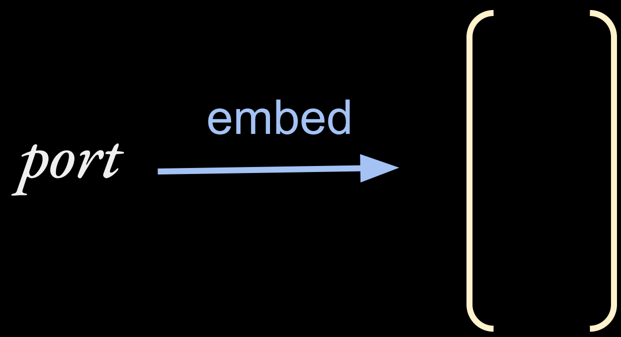
Objective

To embed: convert a token (or sequence) to a vector that **represents meaning**.

Objective

To embed: convert a token (or sequence) to a vector that represents meaning, or is useful to perform downstream NLP application.

Objective



Objective

$$port \xrightarrow{\text{embed}} \begin{pmatrix} 0 \\ \cdots \\ 0 \\ 1 \\ \cdots \\ 0 \end{pmatrix}$$

Objective

one-hot is sparse vector

$$\textit{port} \xrightarrow{\text{embed}} \begin{bmatrix} 0 \\ \cdots \\ 0 \\ 1 \\ \cdots \\ 0 \end{bmatrix}$$

Prefer dense vectors

- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms

For deep learning, in practice, they work better. Why? Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.

Objective

one-h

Prefer dense vectors

- Less parameters (weights) for

el.

implicitly.

s

ce, they work

rameters

nt when you are

ghts rather than

(Jurafsky, 2012)

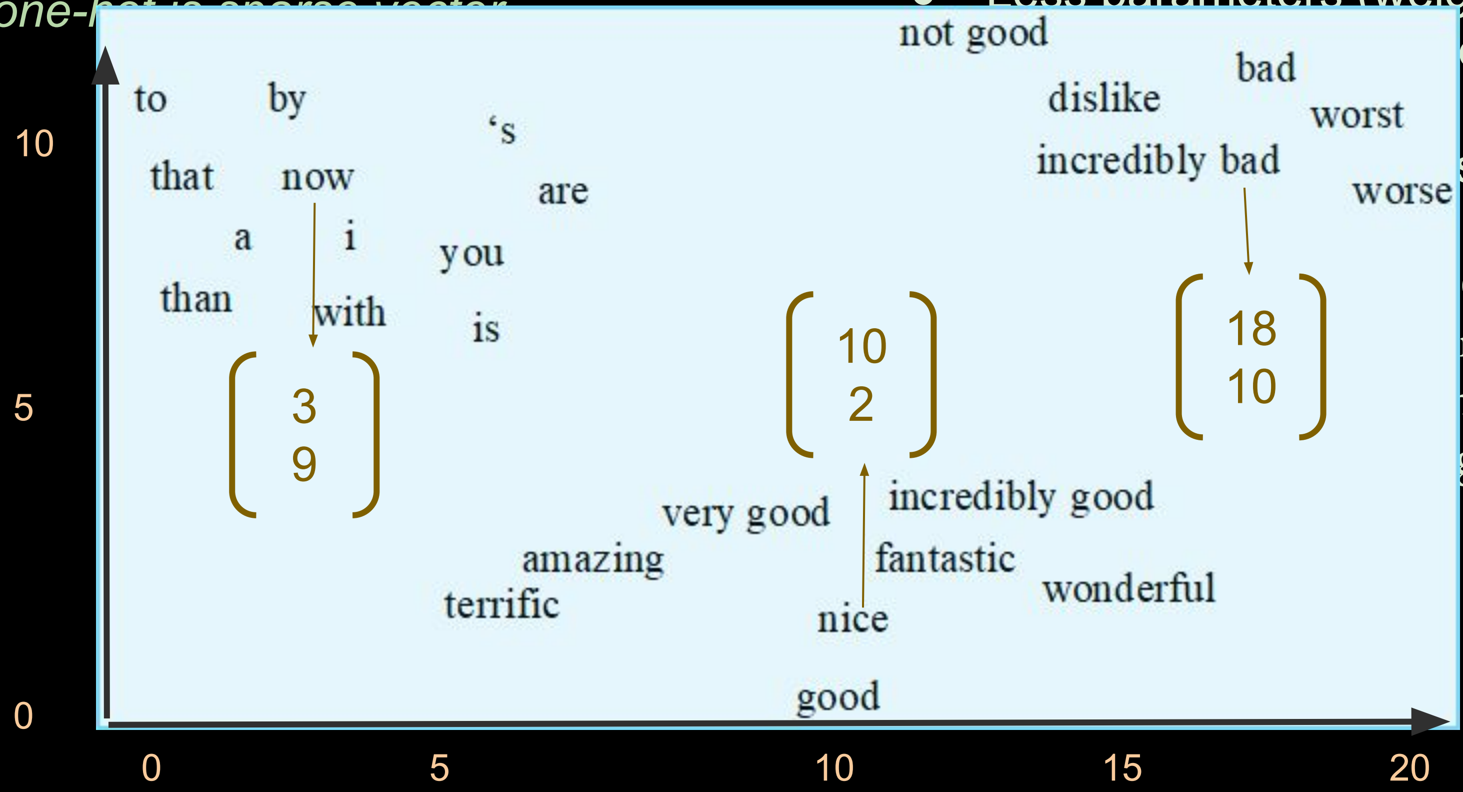
Objective

one-h

Prefer dense vectors

- Less parameters (weights) for
  el.
  implicitly.
  s

  ce, they work
  rameters
  nt when you are
  ghts rather than



(Jurafsky, 2012)

Objective

To embed: convert a token (or sequence) to a vector that represents **meaning.**

Distributional hypothesis

-- A word's meaning is defined by all the different contexts it appears in (i.e. how it is "distributed" in natural language).

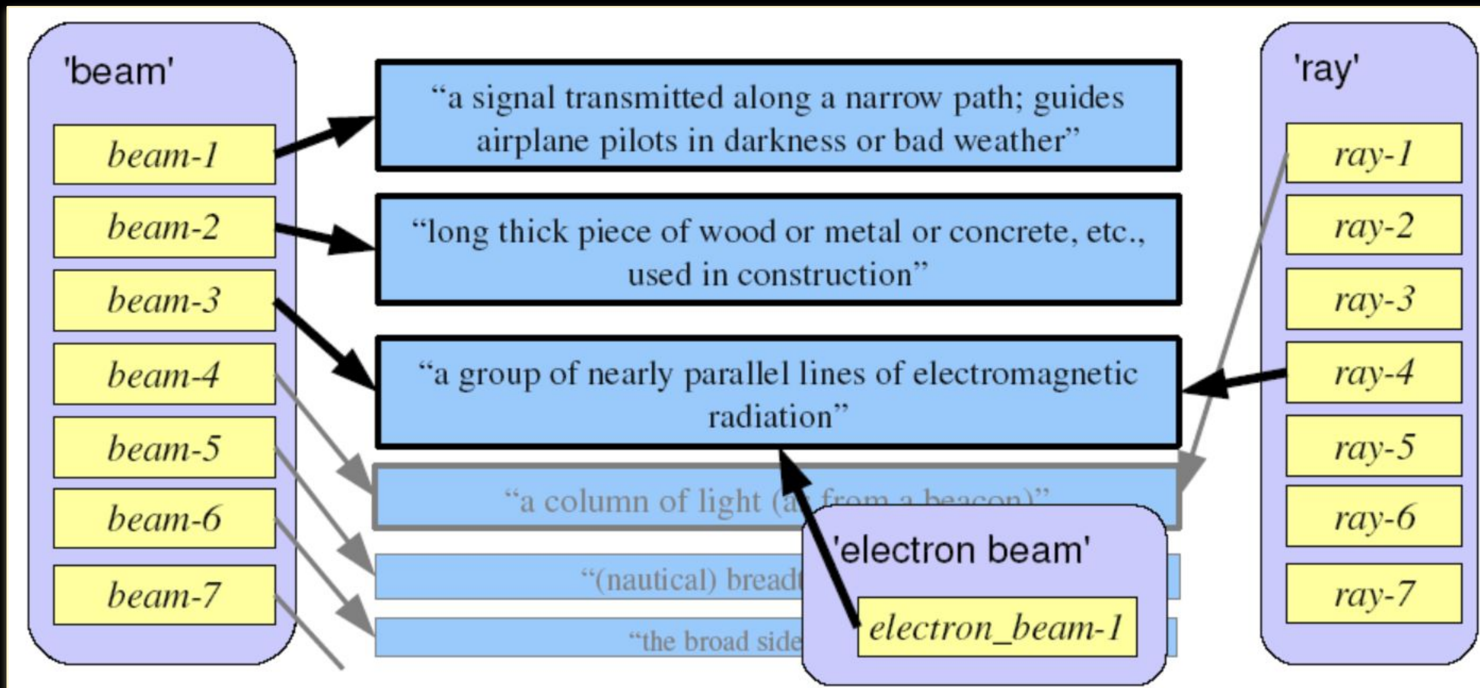
To embed: convert a token (or sequence) to a vector that represents **meaning.**

Wittgenstein, 1945: "*The meaning of a word is its use in the language*"

Firth, 1957: "*You shall know a word by the company it keeps*"

The nail hit the beam behind the wall.

Distributional Hypothesis



The nail hit the beam behind the wall.

Word Vectors

"one-hot encoding"

$$\textit{port} \xrightarrow{\text{embed}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Prefer dense vectors

- Less parameters (weights) for machine learning model.
- May generalize better implicitly.
- May capture synonyms

For deep learning, in practice, they work better. Why? Roughly, less parameters becomes increasingly important when you are learning multiple layers of weights rather than just a single layer.

Word Vectors

"vector embedding"

$$\textit{port} \xrightarrow{\text{embed}} \begin{pmatrix} 0.53 \\ 1.5 \\ 3.21 \\ -2.3 \\ .76 \end{pmatrix}$$

Objective

$$\textit{port} \xrightarrow{\text{embed}} \begin{bmatrix} 0.53 \\ 1.5 \\ 3.21 \\ -2.3 \\ .76 \end{bmatrix}$$

port.n.1 (a place (seaport or airport) where people and merchandise can enter or leave a country)

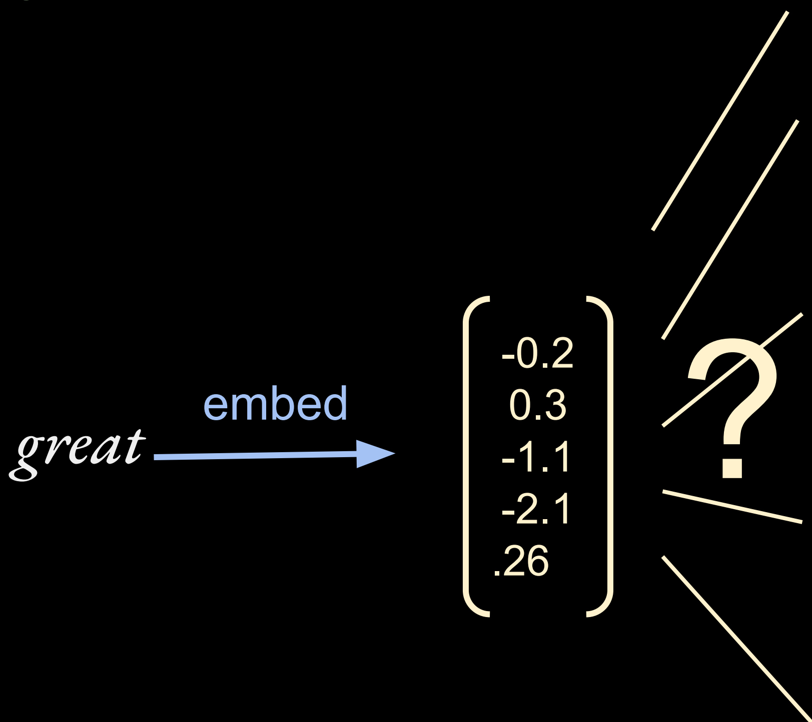
port.n.2 port wine (sweet dark-red dessert wine originally from Portugal)

port.n.3, embrasure, porthole (an opening (in a wall or ship or armored vehicle) for firing through)

larboard, **port**.n.4 (the left side of a ship or aircraft to someone who is aboard and facing the bow or nose)

interface, **port**.n.5 ((computer science) computer circuit consisting of the hardware and associated circuitry that links one device with another (especially a computer and a hard disk drive or other peripherals))

Objective



great.a.1 (relatively large in size or number or extent; larger than others of its kind)

great.a.2, outstanding (of major significance or importance)

great.a.3 (remarkable or out of the ordinary in degree or magnitude or effect)

bang-up, bully, corking, cracking, dandy, **great.a.4**, groovy, keen, neat, nifty, not bad, peachy, slap-up, swell, smashing, old (very good)

capital, **great.a.5**, majuscule (uppercase)

big, enceinte, expectant, gravid, **great.a.6**, large, heavy, with child (in an advanced stage of pregnancy)

great.n.1 (a person who has achieved distinction and honor in some field)

Word2Vec

Principle: Predict missing word.

Similar to classification where y = context and x = word.

p(context | word)

Word2Vec

Principle: Predict missing word.

Similar to classification where y = context and x = word.

p(context | word)

To learn, maximize

Word2Vec

Principle: Predict missing word.

Similar to classification where y = context and x = word.

p(context | word)

To learn, maximize.
In practice, minimize

J = 1 - p(context | word)

Word2Vec: Context

p(context | word)

2 Versions of Context:

1. Continuous bag of words (CBOW): Predict word from context
2. Skip-Grams (SG): predict context words from target

Word2Vec: Context

p(context | word)

2 Versions of Context:

1. Continuous bag of words (CBOW): Predict word from context
2. **Skip-Grams (SG): predict context words from target**

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 embeddings

(Jurafsky, 2017)

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

$$\textit{The nail } \underbrace{\textit{hit}}_{c1}\ \underbrace{\textit{the}}_{c2}\ \underline{\textit{beam}}\ \underbrace{\textit{behind}}_{c3}\ \underbrace{\textit{the}}_{c4}\ \textit{wall.}$$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
...

$$\textit{The nail } \underbrace{\textit{hit}}_{c1}\ \underbrace{\textit{the}}_{c2}\ \underline{\textit{beam}}\ \underbrace{\textit{behind}}_{c3}\ \underbrace{\textit{the}}_{c4}\ \textit{wall.}$$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

The nail hit the beam behind the wall.

c1 c2 c3 c4

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

k negative samples (y=0) for every positive.
How?

The nail $\underbrace{\textit{hit}}_{c1}$ $\underbrace{\textit{the}}_{c2}$ beam $\underbrace{\textit{behind}}_{c3}$ $\underbrace{\textit{the}}_{c4}$ wall.

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

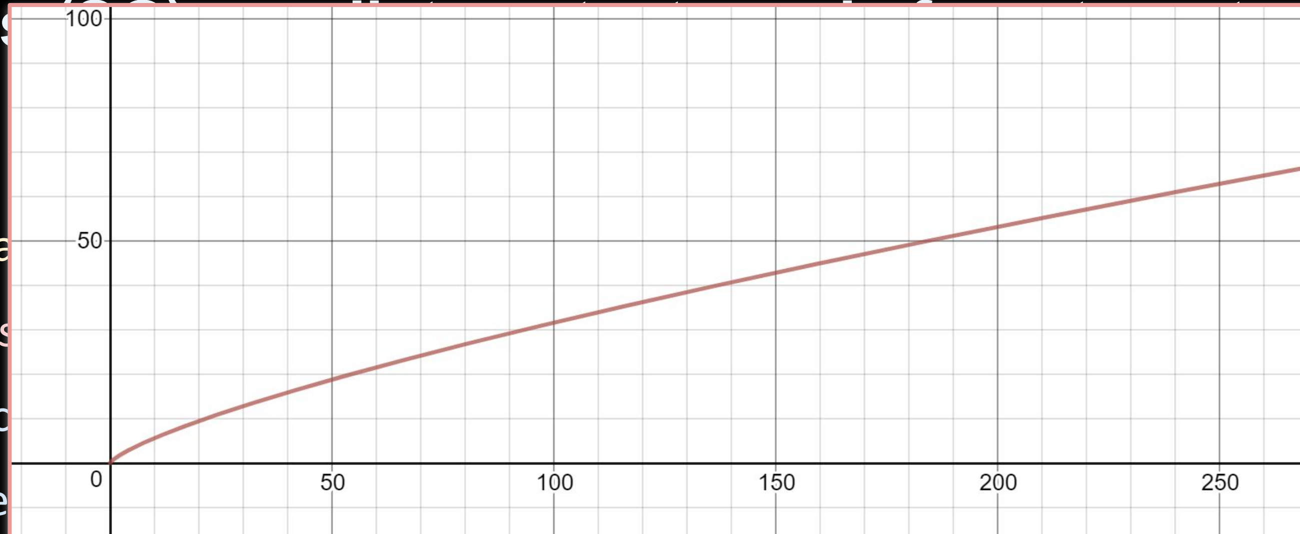
k negative samples (y=0) for every positive.
How? Randomly draw from unigram distribution

$$P(w) = \frac{count(w)}{\sum_w count(w)}$$

The nail hit the beam behind the wall.

c1 c2 c3 c4

Skip-Gram



Steps:

1. Treat the ta
2. Randomly s
3. Use logistic
4. Use the we

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

k negative samples (y=0) for every positive.
How? Randomly draw from unigram distribution, αdjusted

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

where
$\alpha = 0.75$

The nail hit the beam behind the wall.

$\underbrace{\text{hit}}_{c1}\ \underbrace{\text{the}}_{c2}\ \underline{\text{beam}}\ \underbrace{\text{behind}}_{c3}\ \underbrace{\text{the}}_{c4}$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

single context:

$$P(y=1 | c, t) = \frac{1}{1 + e^{-t \cdot c}}$$

$$\textit{The nail } \underbrace{\textit{hit}}_{c1}\ \underbrace{\textit{the}}_{c2}\ \underline{\textit{beam}}\ \underbrace{\textit{behind}}_{c3}\ \underbrace{\textit{the}}_{c4}\ \textit{wall.}$$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

single context:

$$P(y=1|\, c, t) = \frac{1}{1+e^{-t\cdot c}}$$

all contexts

$$P(y=1|\, c, t) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

The nail hit the beam behind the wall.

c1 c2 c3 c4

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

Intuition: t·c is a measure of similarity.

single context:

$$P(y{=}1 \mid c, t) = \frac{1}{1+e^{-t\cdot c}}$$

all contexts

$$P(y{=}1 \mid c, t) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

nail hit the beam behind the wall.

c1 c2 c3 c4

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

Intuition: t·c is a measure of similarity:

But, it is not a probability! To make it one, apply logistic activation:

$$\sigma(z) = 1 / (1 + e^{-z})$$

single context:

$$P(y{=}1 \mid c, t) = \frac{1}{1 + e^{-t \cdot c}}$$

all contexts

$$P(y{=}1 \mid c, t) = \prod_{i=1}^{k} \frac{1}{1 + e^{-t \cdot c_i}}$$

$\textit{e nail}\ \underbrace{\textit{hit}}_{c1}\ \underbrace{\textit{the}}_{c2}\ \underline{\textit{beam}}\ \underbrace{\textit{behind}}_{c3}\ \underbrace{\textit{the}}_{c4}\ \textit{wall.}$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings**

all contexts

$$P(y=1|\ c, t) = \prod_{i=1}^{\kappa} \frac{1}{1+e^{-t\cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

The nail $\underbrace{\textit{hit}}_{c1}$ $\underbrace{\textit{the}}_{c2}$ beam $\underbrace{\textit{behind}}_{c3}$ $\underbrace{\textit{the}}_{c4}$ wall.

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings**

all contexts

$$P(y=1 | c, t) = \prod_{i=1}^{k} \frac{1}{1+e^{-t\cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

$$\textit{The nail } \underbrace{\textit{hit}}_{c1}\ \underbrace{\textit{the}}_{c2}\ \underline{\textit{beam}}\ \underbrace{\textit{behind}}_{c3}\ \underbrace{\textit{the}}_{c4}\ \textit{wall.}$$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

all contexts

$$P(y=1|\, c, t) = \prod_{i=1}^{\kappa} \frac{1}{1+e^{-t \cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
...

3a. assume *dim* * |vocab| weights for each of c and t,

The nail $\underbrace{\text{hit}}_{c1}$ $\underbrace{\text{the}}_{c2}$ beam $\underbrace{\text{behind}}_{c3}$ $\underbrace{\text{the}}_{c4}$ wall.

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

all contexts

$$P(y=1|\ c, t) = \prod_{i=1}^{\kappa} \frac{1}{1+e^{-t \cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

3a. assume *dim* * |vocab| weights for each of c and t, initialized to random values (e.g. *dim = 50* or *dim = 300*)

3b.

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

all contexts

$$P(y=1 \mid c, t) = \prod_{i=1}^{k} \frac{1}{1+e^{-t \cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

3a. assume *dim* * |vocab| weights for each of c and t, initialized to random values (e.g. *dim = 50* or *dim = 300*)

3b. optimize loss:

$$-\sum_{(c,t)} (y) log\, P(y=1|c,t) + (1-y) log\, P(y=0|c,t)$$

Skip-Grams (SG): predict context words from target

p(context | word)

Steps:

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 embeddings

all contexts

$$P(y{=}1|\ c, t) = \prod_{i=1}^{\kappa} \frac{1}{1+e^{-t\cdot c_i}}$$

x = (hit, beam), y = 1
x = (the, beam), y = 1
x = (behind, beam), y = 1
…
x = (happy, beam), y = 0
x = (think, beam), y = 0
…

3a. assume *dim* * |vocab| weights for each of c and t, initialized to random values (e.g. *dim = 50* or *dim = 300*)

3b. optimize loss:

$$-\sum_{(c,t)} (y) log\ P(y=1|c,t) + (1-y) log\ P(y=0|c,t)$$

Maximizes similarity of (c, t) in positive data (y = 1)

Minimizes similarity of (c, t) in negative data (y = 0)

W2V uses the same multi-class loss function as LogReg!

Logistic Regression Likelihood: $L(\beta_0, \beta_1, \dots, \beta_k | X, Y) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}$

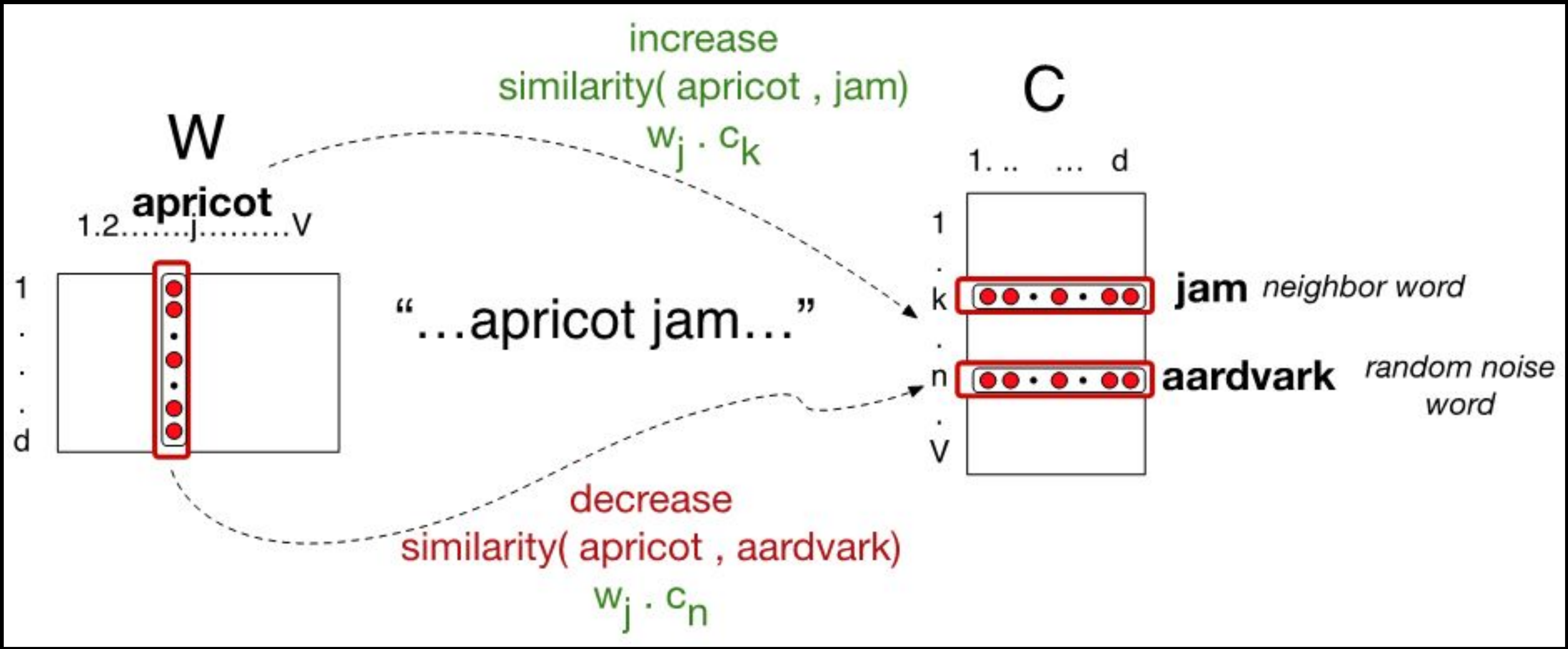
Log Likelihood: $\ell(\beta) = \sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p(x_i))$

Log Loss: $J(\beta) = -\frac{1}{N}\sum_{i=1}^{N} y_i log\, p(x_i) + (1 - y_i) log\, (1 - p)(x_i))$

Cross-Entropy Cost: $J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{|V|} y_{i,j} log\, p(x_{i,j})$ (a “multiclass” log loss)

In vector algebra form: `- mean( sum( y*log(y_pred) ) )`

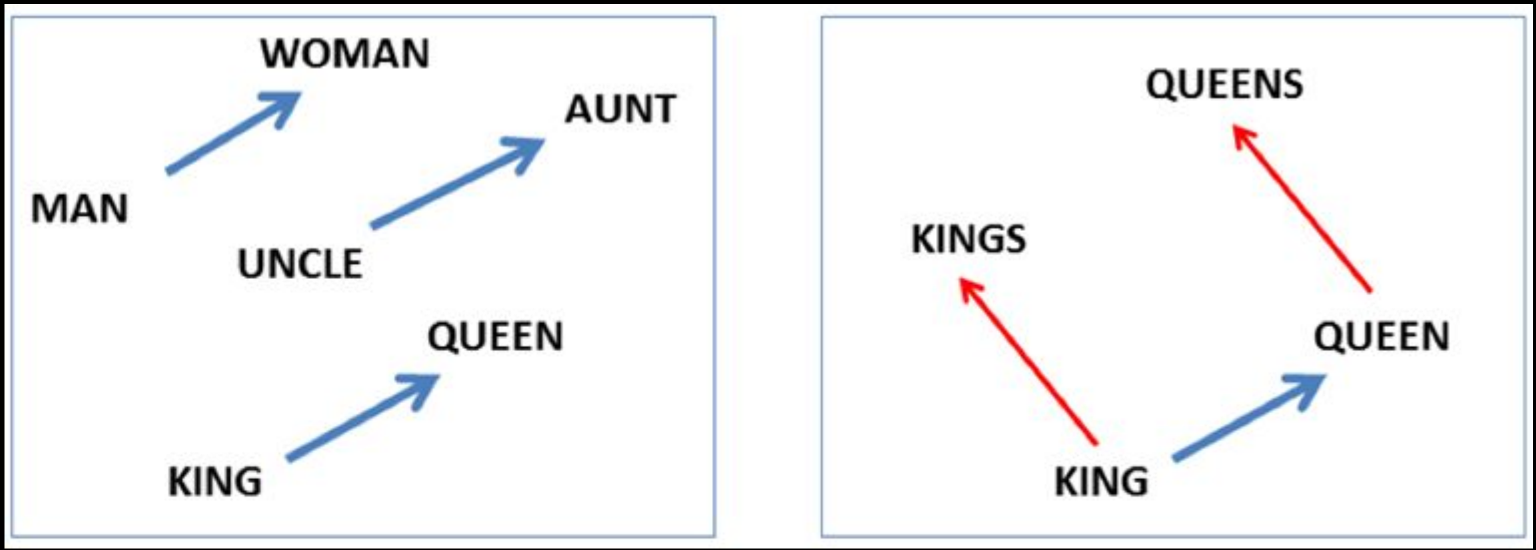
Word 2 Vec



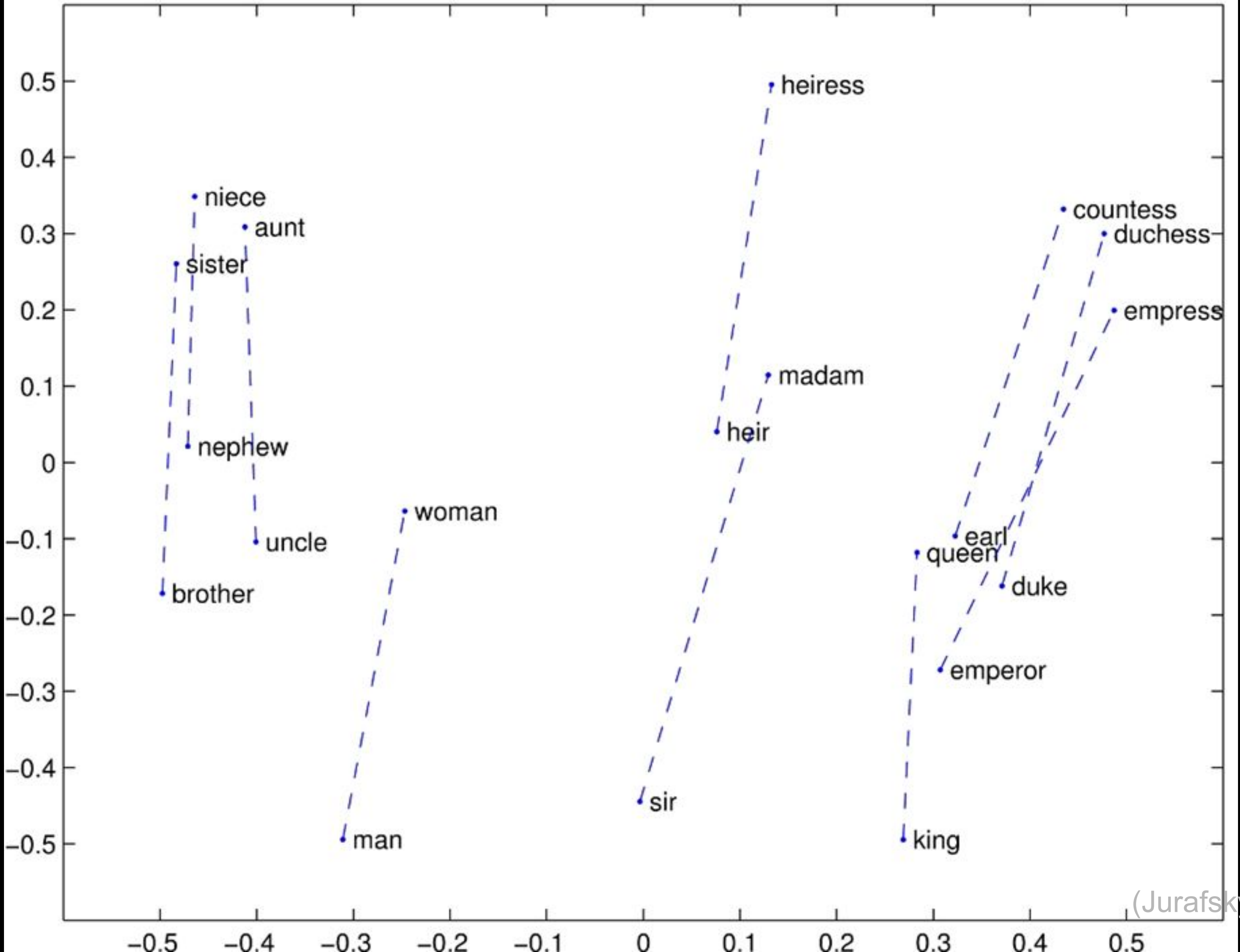
$$\sum_{(c,t)} (y) log P(y = 1|c, t) + (y - 1) log P(y = 0|c, t)$$

(Jurafsky, 2017)

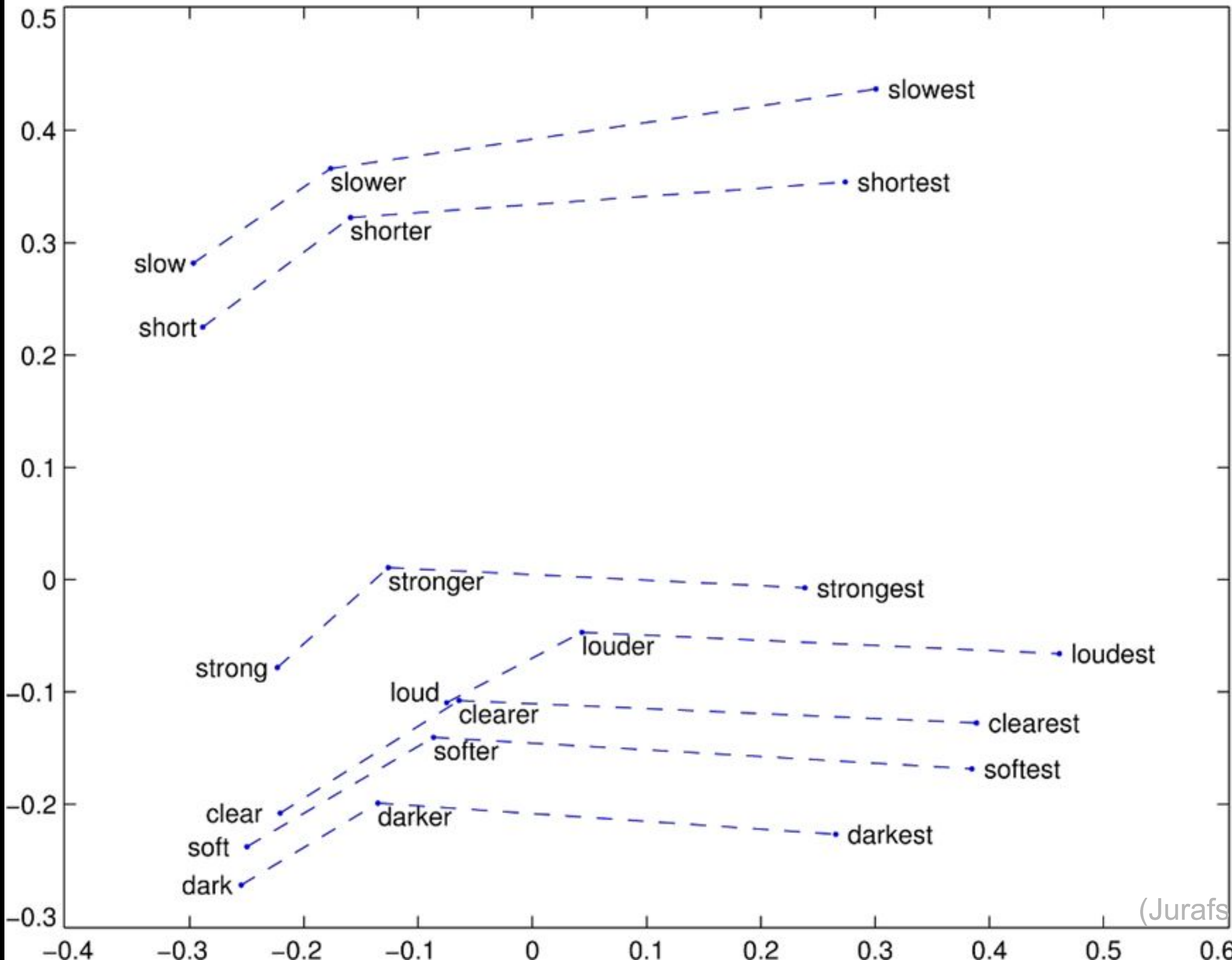
Word2Vec captures analogies (kind of)



(Jurafsky, 2017)



(Jurafsky, 2017)



(Jurafsky, 2017)

Word2Vec: Quantitative Evaluations

1. Compare to manually annotated pairs of words: WordSim-353 (Finkelstein et al., 2002)

2. Compare to words in context (Huang et al., 2012)

3. Answer TOEFL synonym questions.

Word2Vec: Quantitative Evaluations

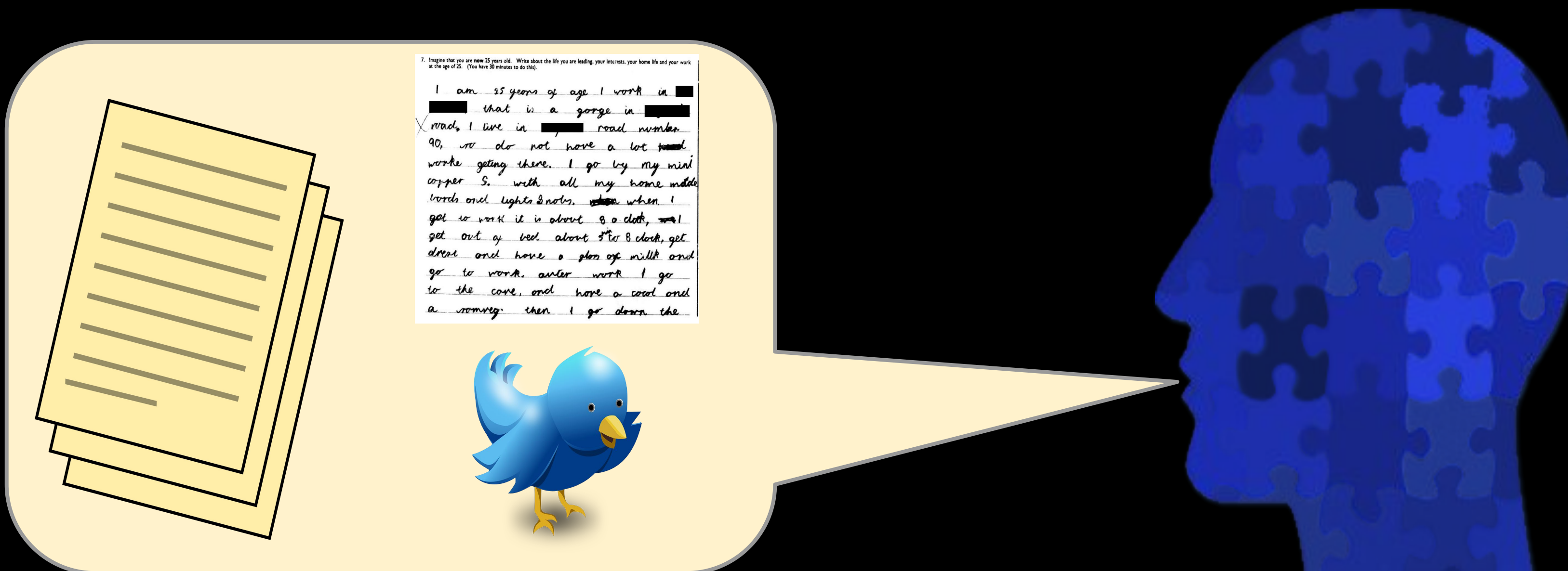
1. Compare to manually annotated pairs of words: WordSim-353 (Finkelstein et al., 2002)

2. Compare to words in context (Huang et al., 2012)

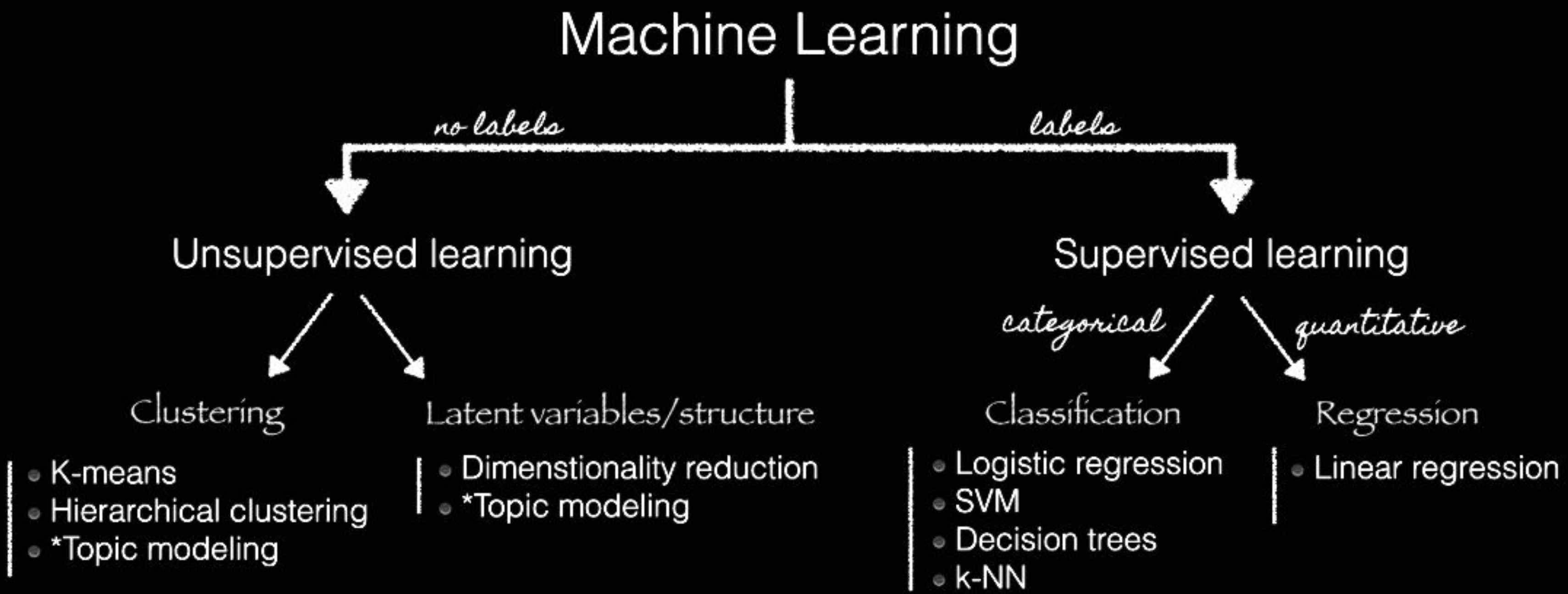
3. Answer TOEFL synonym questions.

What have we learned since Word2vec? (a lot, but here are 2 important points)

1. Improved loss function: GlOVE embeddings (Pennington et al., 2014)
2. Word2Vec itself performs very similarly to PCA on a co-occurrence matrix ("LSA" Deerwater et al., 1988 – a much much older techniques!).


7. Imagine that you are **now** 25 years old. Write about the life you are leading, your interests, your home life and your work at the age of 25. (You have 30 minutes to do this).
I am 25 yeors of age I work in [redacted] that is a gorge in [redacted] road, I live in [redacted] road number 90, we do not hove a lot worke geting there. I go by my mini copper S. with all my home made borch and lights &nobs. when I got to work it is about 8 o clock, I get out of bed about 5 min to 8 clock, get drest and hove a glass of millk and go to work. aufter work I go to the cove, and hove a cocol and a somveg. then I go down the

Topic Modeling



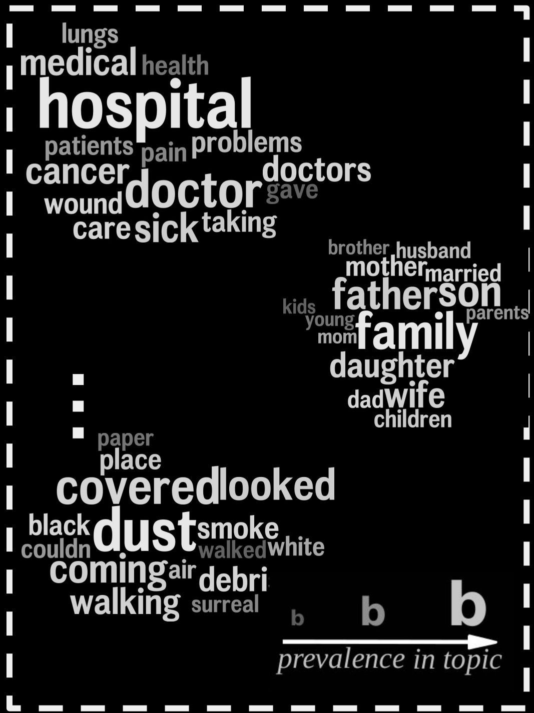
(Doig, 2014)

Topic Modeling

Topic: A group of highly related words and phrases. (aka "semantic field")

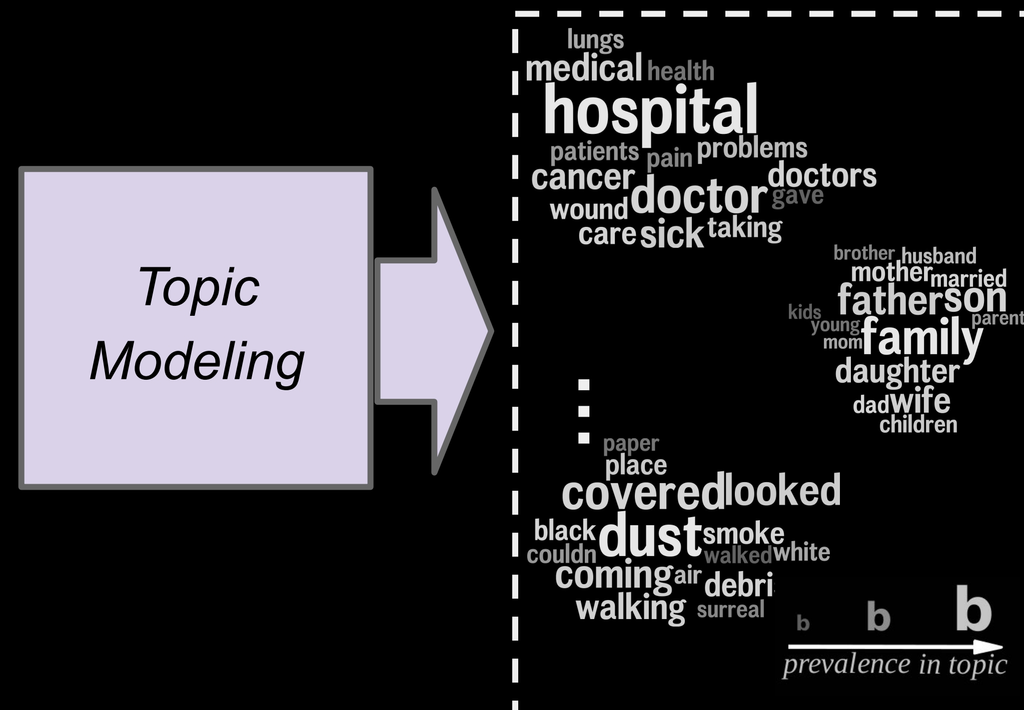
example: from WTC responder interviews

(Son et al., 2021)



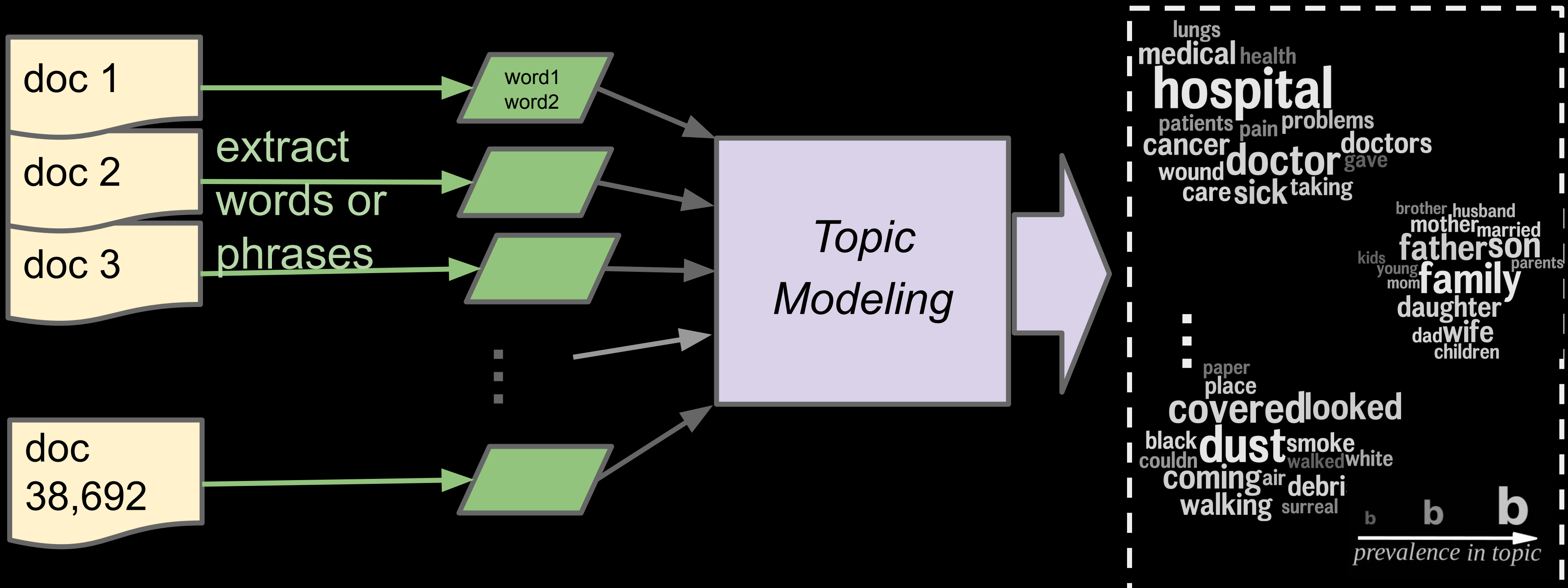
Topic Modeling

Topic: A group of highly related words and phrases. (aka "semantic field")



Topic Modeling

Topic: A group of highly related words and phrases. (aka "semantic field")



Select Example Topics

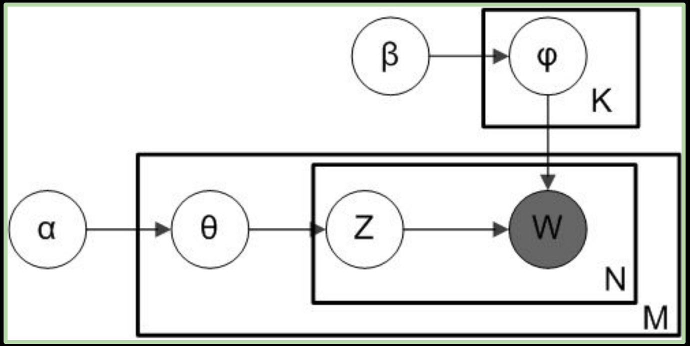
Generating Topics from Documents

- *Latent Dirichlet Allocation* -- a Bayesian probabilistic model where by words which appear in similar *contexts* (i.e. in essays that have similar sets of words) will be clustered into a prespecified number of topics.
- Rule of thumb: $|\text{topics}| = \dfrac{|\text{observations}|}{100}$
- Each document receives a score per topic -- a probability: $p(\text{topic}|\text{doc})$.

Doc 1	Doc 2	Doc 3
topic 1: .05	topic 1: .03	topic 1: .04
topic 2: .02	topic 2: .01	topic 2: .03
topic 3: .01	topic 3: .03	topic 3: .03
…	…	…
topic 100: .07	topic 100: .05	topic 100: .06

Latent Dirichlet Allocation
(Blei et al., 2003)

- LDA specifies a Bayesian probabilistic model where by
  - documents are viewed as a distribution of topics,
  - topics are a distribution of words.



Observed:

W -- observed word in document *m*

Inferred:

θ -- topic distribution for document *m*,

Z -- topic for word *n* in document *m*

φ --word distribution for topic *k*

Priors

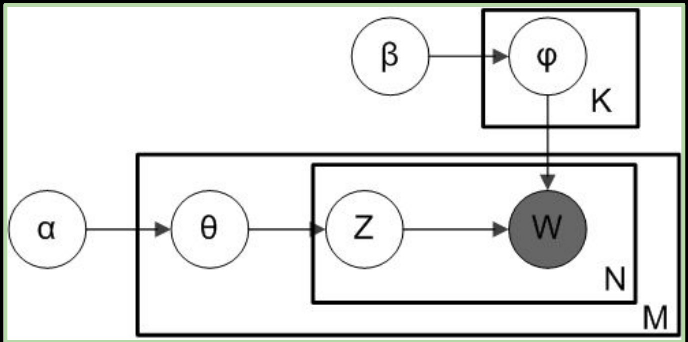
α -- hyperparameter for Dirichlet prior on the **topics per document.**

β -- hyperparameter for Dirichlet prior on the **words per topic**.

K – number of topics

Latent Dirichlet Allocation
(Blei et al., 2003)

- LDA specifies a Bayesian probabilistic model where by documents are viewed as a distribution of topics, and topics are a distribution of words.
- How to estimate (i.e. fit) the model parameters given data and priors? Common choices:
  - Gibb's Sampling (best)
  - variational Bayesian Inference (fastest).
- Key Output: the "posterior" $\varphi = p(word \mid topic)$, the probability of a word given a topic.
  From this and p(topic), we can get: $p(topic|word)$

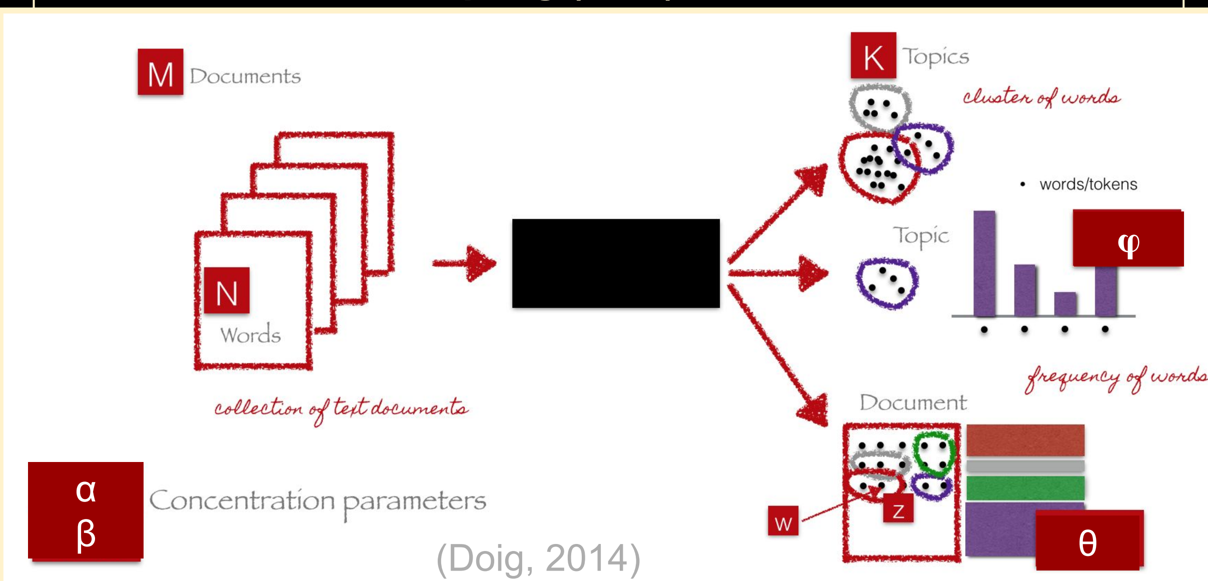


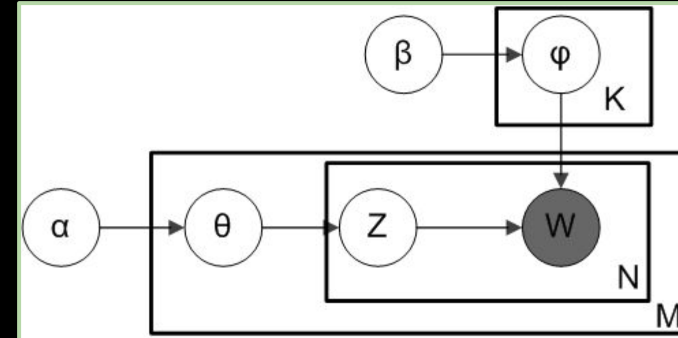
Observed:
W -- observed word in document *m*
Inferred:
θ -- topic distribution for document *m*,
Z -- topic for word *n* in document *m*
φ --word distribution for topic *k*
Priors
α -- hyperparameter for Dirichlet prior on the **topics per document.**
β -- hyperparameter for Dirichlet prior on the **words per topic**.
K – number of topics

Latent Dirichlet Allocation

(Blei et al., 2003)

- LDA specifies a Bayesian probabilistic model where by documents are viewed as a distribution of topics, and topics are a distribution of words.
- How to estimate (i.e. fit) the model parameters given data and priors? Common choices:
    - **Gibb's Sampling (best)**





Observed:

W -- observed word in document *m*

Inferred:

θ -- topic distribution for document *m*,

Z -- topic for word *n* in document *m*

φ --word distribution for topic *k*

Priors

α -- hyperparameter for Dirichlet prior on the **topics per document.**

β -- hyperparameter for Dirichlet prior on the **words per topic**.

K – number of topics

Example

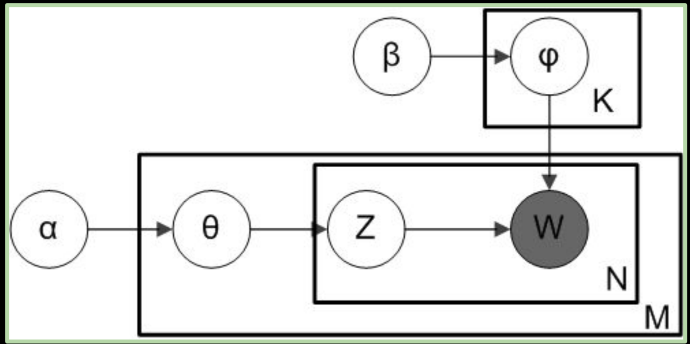
Most prevalent words for 4 topics are listed at the top and words associated with them from a Yelp review are colored accordingly below.

Labor and Delivery	Patient treatment	Surgery/ procedure and peri-op	Insurance and Billing
Baby	Care	Surgery	Insurance
Birth	Staff	Procedure	Billing
Nurses	Nurses	Surgeon	Bill
Labor	Hospital	Recovery	Hospital
Delivery	Doctors	Day	Department
Experience	Great	Staff	Company
Nurse	Caring	Experience	Paid

It depends what you look for in a hospital. Remember that this is a teaching hospital so you must adjust your expectations accordingly. This means many students who, bless their hearts, may ask you the same questions again and again. I waited for hours on standby to deliver my baby by emergency c-section. The kind nurses who served me during recovery and the anesthesiologist on duty during my surgery deserve praise. My OB was very competent, but I wish he were willing to do an extraversion or at least given me an epidural. Im grateful they ultimately did what was best for my kid. However, I think things could have happened a lot more smoothly with better pain control. The only other thing to watch out for is your bills. This is the only institution I have been to that bills me prior to billing insurance. I fought two years to claim a credit through a database system change. The cafeteria gets flack for being all vegetarian but you just have to know what to order. Stay there for 1-2 weeks and you get the hang of whats good and whats not.

Ranard, B.L., Werner, R.M., Antanavicius, T., Schwartz, H.A., Smith, R.J., Meisel, Z.F., Asch, D.A., Ungar, L.H. & Merchant, R.M. (2016). Yelp Reviews Of Hospital Care Can Supplement And Inform Traditional Surveys Of The Patient Experience Of Care. *Health Affairs, 35(4),* 697-705.

Latent Dirichlet Allocation
(Blei et al., 2003)



- LDA specifies a Bayesian probabilistic model where by documents are viewed as a distribution of topics, and topics are a distribution of words.
- How to estimate (i.e. fit) the model parameters given data and priors? Common choices:
  - Gibb's Sampling (best)
  - variational Bayesian Inference (fastest).
- Key Output: the "posterior" $\varphi = p(word \mid topic)$, the probability of a word given a topic.

  From this and p(topic), we can get: $p(topic|word)$

Observed:

W -- observed word in document *m*

Inferred:

θ -- topic distribution for document *m*,

Z -- topic for word *n* in document *m*

φ --word distribution for topic *k*

Priors

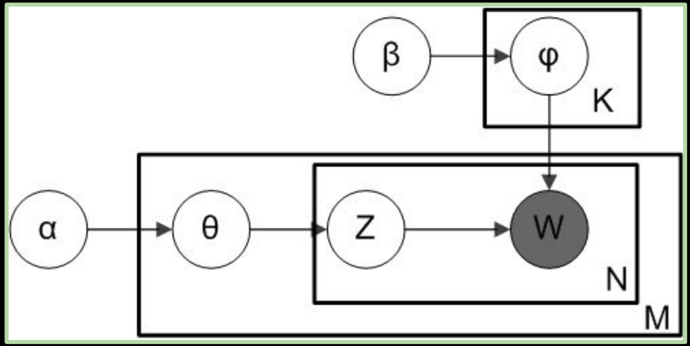
α -- hyperparameter for Dirichlet prior on the **topics per document.**

β -- hyperparameter for Dirichlet prior on the **words per topic**.

K – number of topics

Latent Dirichlet Allocation

(Blei et al., 2003)



- LDA specifies a Bayesian probabilistic model where by documents are viewed as a distribution of topics, and topics are a distribution of words.
- How to estimate (i.e. fit) the model parameters given data and priors? Common choices:
  - Gibb's Sampling (best)
  - variational Bayesian Inference (fastest).
- Key Output: the "posterior" $\varphi = p(word \mid topic)$, the probability of a word given a topic.

  From this and p(topic), we can get: $p(topic|word)$

To Apply:

$$p(topic|doc) = \sum_{word \in topic} p(topic|word)p(word|doc)$$

Observed:

W -- observed word in document *m*

Inferred:

θ -- topic distribution for document *m*,

Z -- topic for word *n* in document *m*

φ --word distribution for topic *k*

Priors

α -- hyperparameter for Dirichlet prior on the **topics per document.**

β -- hyperparameter for Dirichlet prior on the **words per topic**.

K – number of topics

Topic Modeling Packages

Most Reliable: Mallet (Java; uses Gibb's Sampling),
pymallet (slower than Mallet but high quality results)

Ease of use: Gensim (python; uses variational inference;
implements word2vec as well)

Topic Modeling

Common applications:

- **Open vocabulary content analysis:** Describing the latent semantic categories of words or phrases present across a set of documents

- **Embeddings for predictive task:** for all topics, use p(topic|document) as score. Feed to predictive model (e.g. classifier).

PCA-Based Embeddings

Dimensionality reduction
-- try to represent with only p' dimensions

also known as "Latent Semantic Analysis"

Supplement: SVD Implementation details not within scope but the concept of using PCA on word co-occurrence matrix was covered.

PCA-Based Embeddings

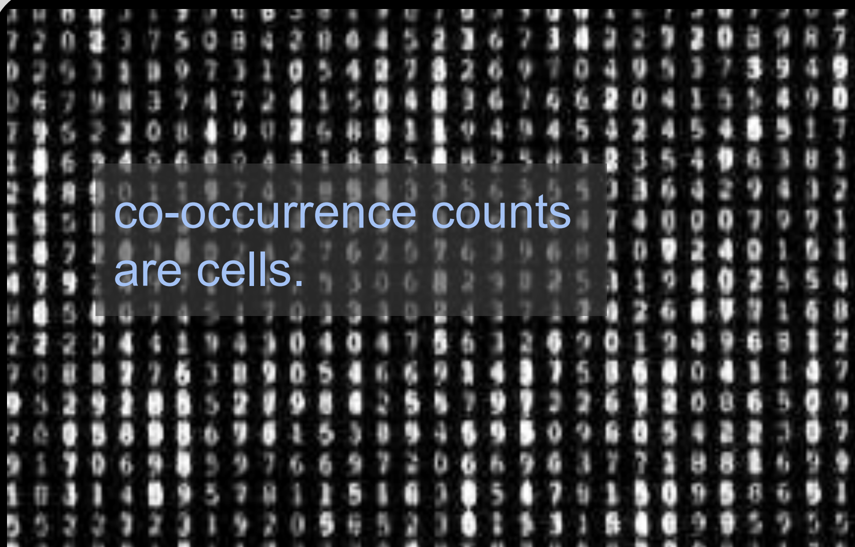
Dimensionality reduction
-- try to represent with only p' dimensions

also known as "Latent Semantic Analysis"

context words are features

w1, w2, w3, w4, … wp

w1
w2
w3
…
w_n



target words are observations

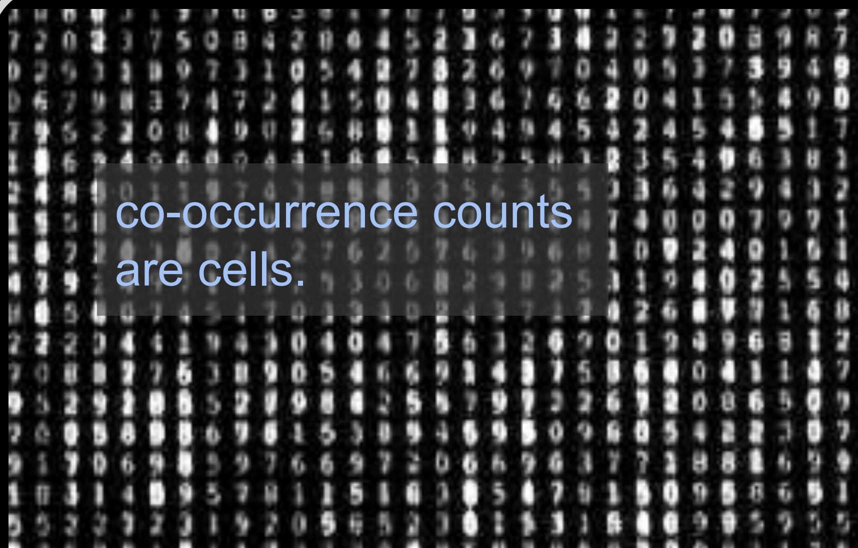
PCA-Based Embeddings

Dimensionality reduction
-- try to represent with only p’ dimensions
p' < p

context words are features

w1, w2, w3, w4, … wp

w1
w2
w3
…
w_n

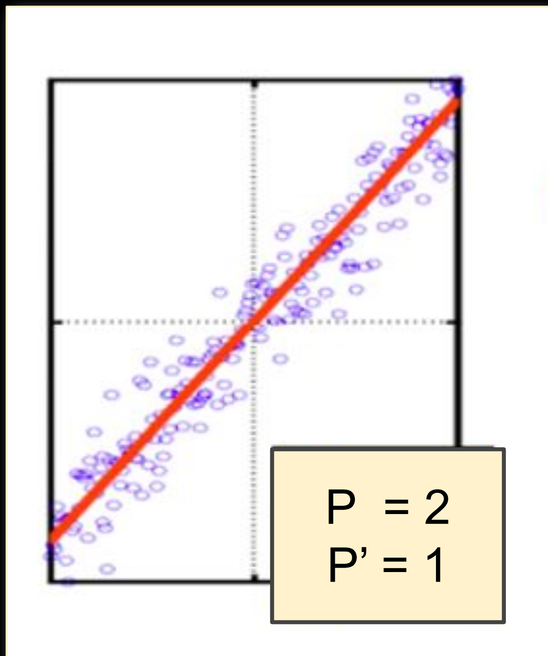


target words are observations

c1, c2, c3, c4, … cp’

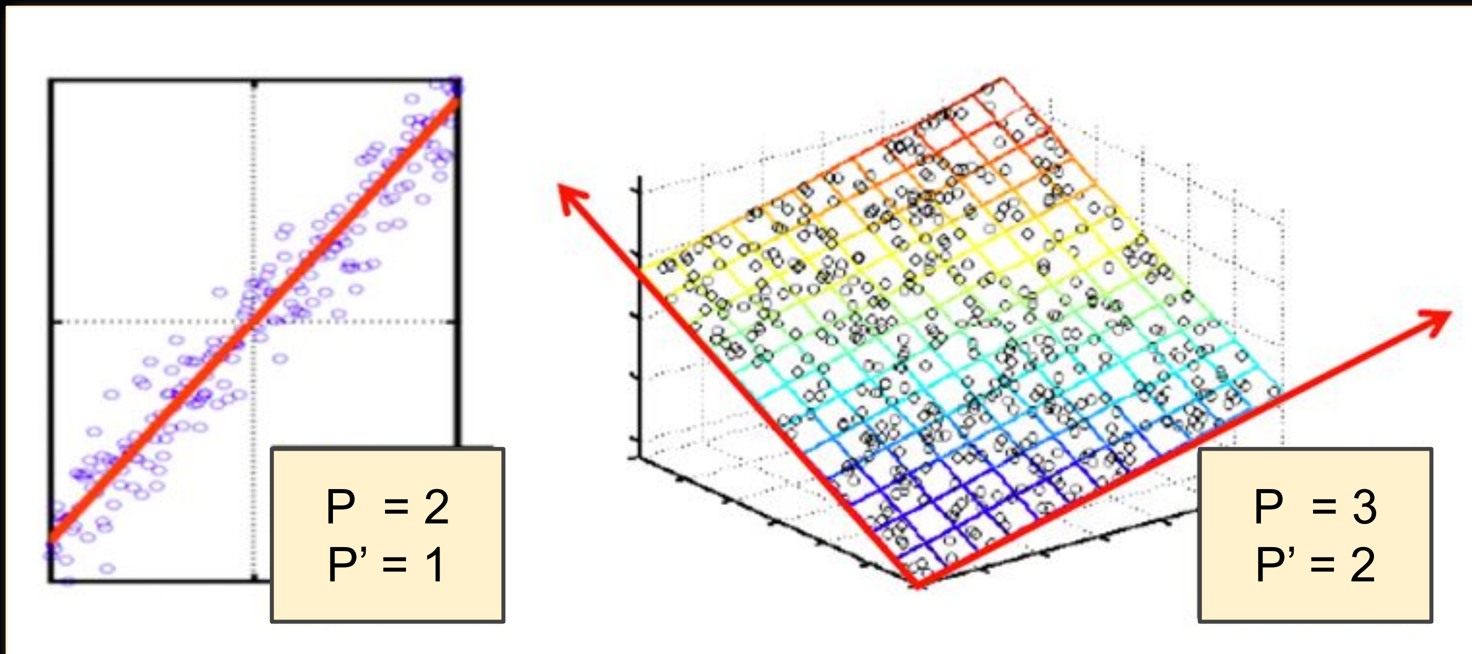
w1
w2
w3
…
w_n

Concept: Dimensionality Reduction in 3-D, 2-D, and 1-D



Data (or, at least, what we want from the data) may be accurately represented with less dimensions.

Concept: Dimensionality Reduction in 3-D, 2-D, and 1-D



Data (or, at least, what we want from the data) may be accurately represented with less dimensions.

Concept: Dimensionality Reduction

Rank: Number of linearly independent columns of A.
(i.e. columns that can't be derived from the other columns through addition).

Q: How many columns do we really need?

$$\begin{pmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{pmatrix}$$

Concept: Dimensionality Reduction

Rank: Number of linearly independent columns of A.
(i.e. columns that can’t be derived from the other columns through addition).

Q: How many columns do we really need?

$$\begin{pmatrix} 1 & -2 & 3 \\ 2 & -3 & 5 \\ 1 & 1 & 0 \end{pmatrix}$$

A: 2. The 1st is just the sum of the second two columns

… we can represent as linear combination of 2 vectors:

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -2 \\ -3 \\ 1 \end{bmatrix}$$

SVD-Based Embeddings

Dimensionality reduction
-- try to represent with only p’ dimensions

context words are features

f1, f2, f3, f4, … fp

o1
o2
o3
…
o_n

target words are observations

c1, c2, c3, c4, … cp’

o1
o2
o3
…
o_n

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^{T}$$

X: original matrix, U: “left singular vectors”,

D: “singular values” (diagonal), V: “right singular vectors”

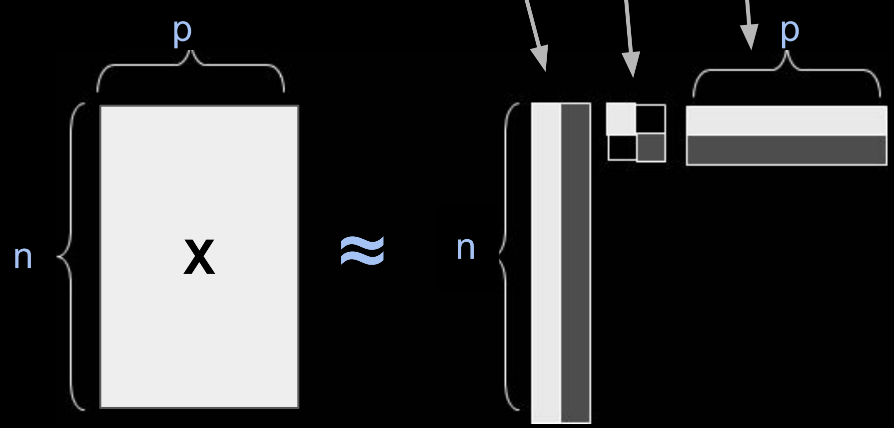
Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} = U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix,
U: “left singular vectors”,
D: “singular values” (diagonal),
V: “right singular vectors”



Dimensionality Reduction - PCA - Example

$$X_{[n \times p]} = U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

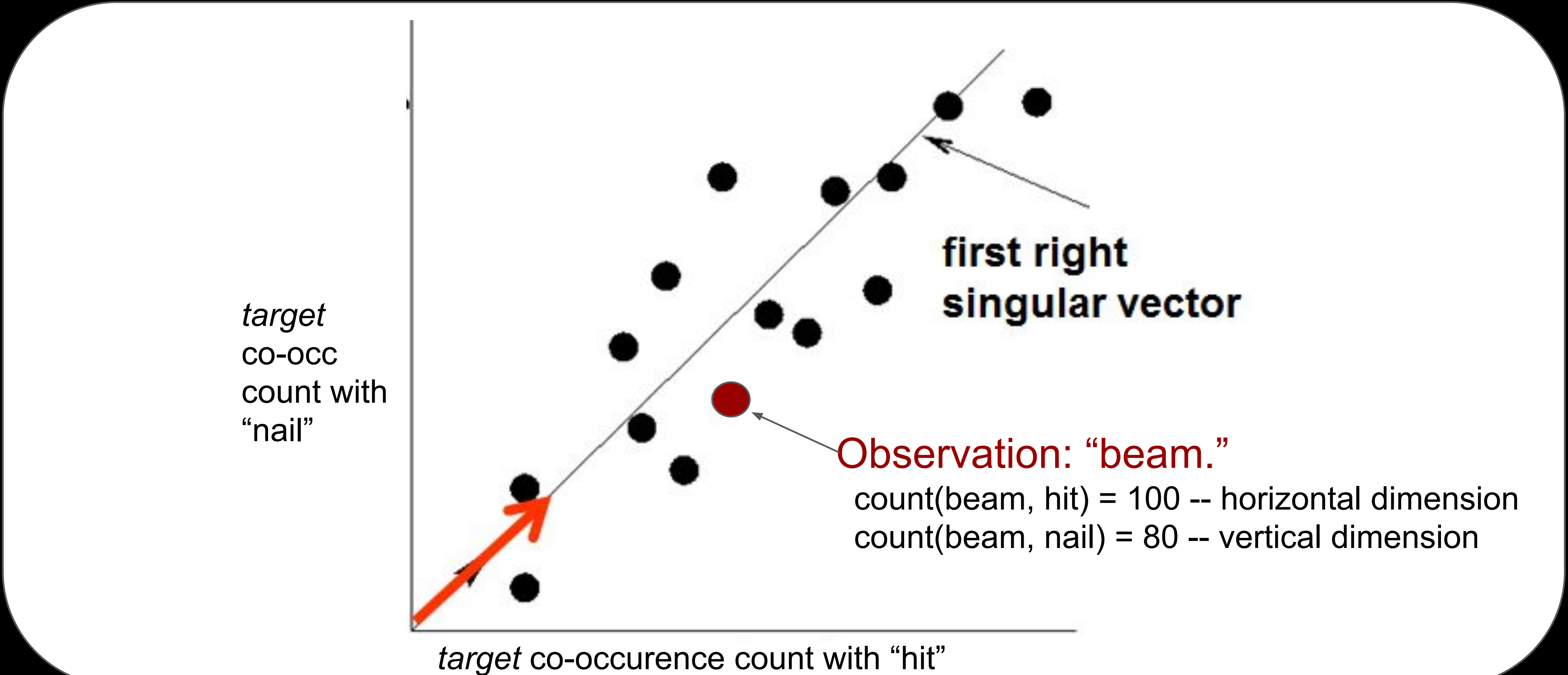
Word co-occurrence counts:

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

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 13

Dimensionality Reduction - PCA - Example

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$



Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^T$$

X: original matrix, U: "left singular vectors",
D: "singular values" (diagonal), V: "right singular vectors"

Projection (dimensionality reduced space) in 3 dimensions:

$$(U_{[nx3]} D_{[3x3]} V_{[px3]}^T)$$

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[n \times p]} \cong U_{[n \times r]} D_{[r \times r]} V_{[p \times r]}^T$$

X: original matrix, U: "left singular vectors",
D: "singular values" (diagonal), V: "right singular vectors"

To check how well the original matrix can be reproduced:

$Z_{[n \times p]} = U D V^T$, How does Z compare to original X?

Dimensionality Reduction - PCA

Li … ns.

:

Goal: Minimize the sum of reconstruction errors:

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|x_{ij}-z_{ij}\right\|^{2}$$

- where x_{ij} are the “old” and z_{ij} are the “new” coordinates

X: original matri … ular vectors”,
D: “singular valu … gular vectors”

To check how well the original matrix can be reproduced:
$Z_{[nxp]} = U\ D\ V^T$, How does Z compare to original X?

The loss function that SVD solves

Dimensionality Reduction - PCA

Li ... ns.

Goal: Minimize the sum of reconstruction errors:

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\left\|x_{ij} - z_{ij}\right\|^2$$

- where x_{ij} are the "old" and z_{ij} are the "new" coordinates

X: original matri ... ular vectors",
D: "singular valu ... gular vectors"

To check how well the original matrix can be reproduced:

$Z_{[nxp]} = U\, D\, V^T$, How does Z compare to original X?

Dimensionality Reduction - PCA

Linear approximates of data in r dimensions.

Found via *Singular Value Decomposition:*

$$X_{[nxp]} \cong U_{[nxr]} D_{[rxr]} V_{[pxr]}^{T}$$

U, D, and V are unique

D: always positive