

# **SI 630**

# **Natural Language Processing:**

# **Algorithms and People**

## Lecture 4: Word Vectors

Jan. 29, 2019

David Jurgens  
jurgens@umich.edu



*Eat ~ Pray ~ Love*

# Lexical Semantics

# What do words mean?

- First thought: look in a dictionary
- <http://www.oed.com/>

# Words, Lemmas, Senses, Definitions

## pepper, *n.*

Pronunciation: Brit. /'pɛpə/, U.S. /'pɛpər/

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

Frequency (in current use):

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

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

1. The spice or the plant.

1.

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

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

2.

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

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

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

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

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

# Words, Lemmas, Senses, Definitions

## lemma

### pepper, n.

Pronunciation: BRIT. /'pɛpə/, U.S. /'pɛpər/

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

Frequency (in current use):

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

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

1. The spice or the plant.

1.

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

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

2.

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

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

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

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

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

# Words, Lemmas, Senses, Definitions

## lemma

pepper, n.

Pronunciation: BRIT. /'pɛpə/, U.S. /'pɛpər/

Forms: OE *peopor* (rare), OE *pipcer* (transmission error), OE *pirpor*, OE *pirpur* (rare)

Frequency (in current use):

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

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

1. The spice or the plant.

1.  
a. A hot pungent spice derived from the prepared fruits (peppercorns) of the pepper plant, *Piper nigrum* (see sense 2a), used from early times to season food, either whole or ground to powder (often in association with salt). Also (locally, chiefly with distinguishing word): a similar spice derived from the fruits of certain other species of the genus *Piper*; the fruits themselves.

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

2.  
a. The plant *Piper nigrum* (family Piperaceae), a climbing shrub indigenous to South Asia and also cultivated elsewhere in the tropics, which has alternate stalked entire leaves, with pendulous spikes of small green flowers opposite the leaves, succeeded by small berries turning red when ripe. Also more widely: any plant of the genus *Piper* or the family Piperaceae.

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

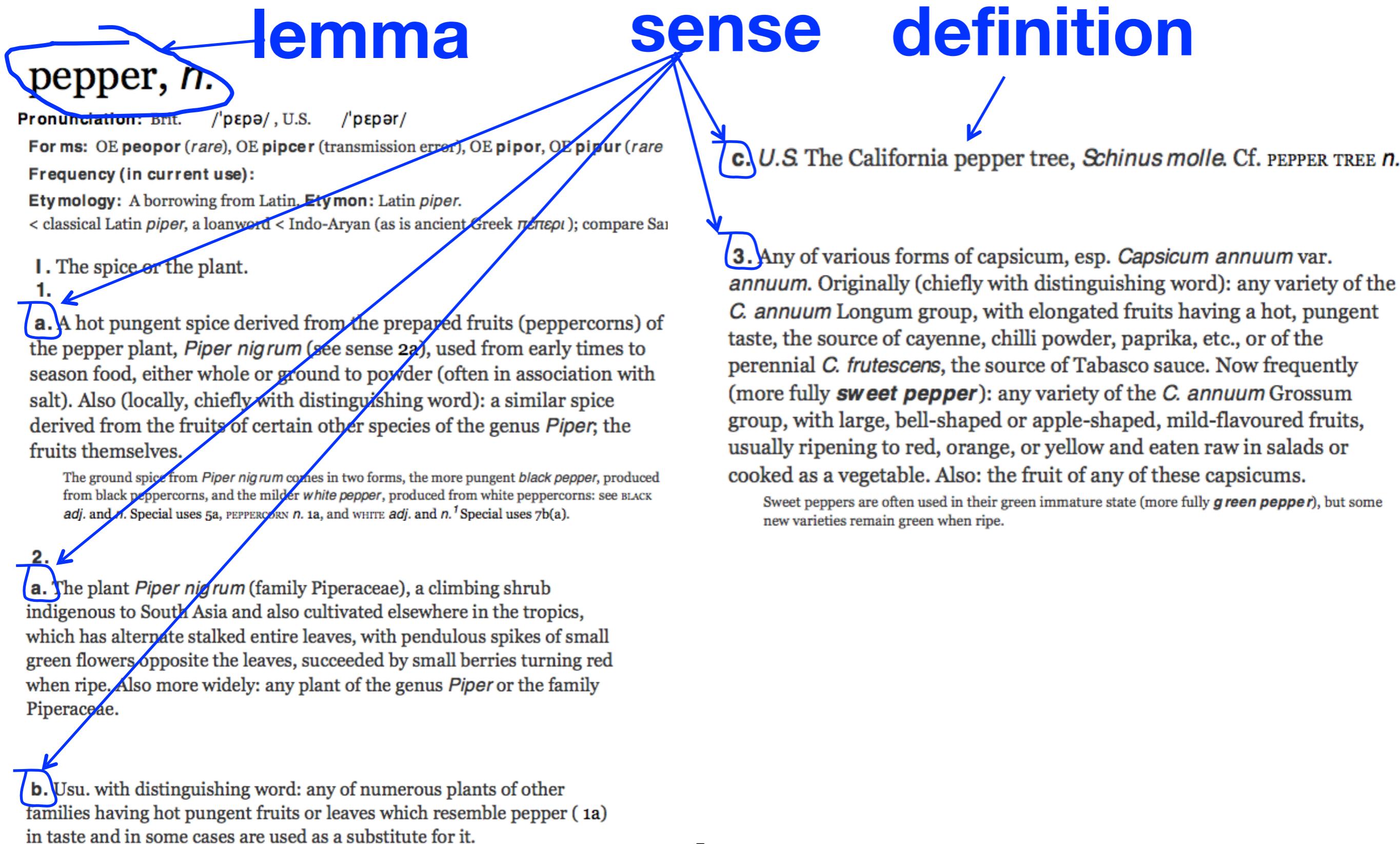
## sense

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

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

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

# Words, Lemmas, Senses, Definitions



# Lemma pepper

Sense 1: spice from pepper plant

# Lemma pepper

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

# Lemma pepper

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

# Lemma pepper

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

Sense 4: another plant with peppercorns  
(California pepper)

# Lemma pepper

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

Sense 4: another plant with peppercorns  
(California pepper)

Sense 5: *capsicum* (i.e. chili, paprika, bell pepper, etc)

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

Sense 1: spice from pepper plant

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

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

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

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

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

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

Sense 4: another plant with peppercorns  
(California pepper)

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

Sense 1: spice from pepper plant

Sense 2: the pepper plant itself

Sense 3: another similar plant (Jamaican pepper)

Sense 4: another plant with peppercorns (California pepper)

Sense 5: *capsicum* (i.e. chili, paprika, bell pepper, etc)

There are relations between  
senses

# Relation: Synonymity

- Synonyms have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - big / large
  - automobile / car
  - vomit / throw up
  - Water / H<sub>2</sub>O

# Relation: Synonymity

- 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.
- The Linguistic Principle of Contrast:
  - Difference in form -> difference in meaning

# Relation: Synonymity?

Water/H<sub>2</sub>O

Big/large

Brave/courageous

# Relation: Antonymy

- Senses that are opposites with respect to one feature of meaning
- Otherwise, they are very similar!

# Relation: Antonymy

- Senses that are opposites with respect to one feature of meaning
- Otherwise, they are very similar!
  - dark/light      short/long      fast/slow      rise/fall
  - hot/cold          up/down                        in/out

# Relation: Antonymy

- Senses that are opposites with respect to one feature of meaning
- Otherwise, they are very similar!
  - dark/light      short/long      fast/slow      rise/fall
  - hot/cold          up/down                        in/out
- More formally: antonyms can
  - define a binary opposition
    - or be at opposite ends of a scale
    - long/short, fast/slow
  - Be *reversives*:
    - rise/fall, up/down

# Relation: Similarity

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

car, bicycle

cow, horse

# Ask humans how similar two words are

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

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

# Relation: Word relatedness

- Also called "word association"
- Words be related in any way, perhaps via a semantic frame or field
  - car, bicycle: similar
  - car, gasoline: related, not similar

# Semantic field

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

**hospitals**

*surgeon, scalpel, nurse, anaesthetic, hospital*

**restaurants**

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

**houses**

*door, roof, kitchen, family, bed*

# Relation: Superordinate/ subordinate

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

- *car* is a subordinate of vehicle
- *mango* is a subordinate of *fruit*

Conversely **superordinate**

- *vehicle* is a superordinate of *car*
- *fruit* is a superordinate of *mango*

<b>Superordinate</b>	vehicle	fruit	furniture
<b>Subordinate</b>	car	mango	chair

# Name these items

# Name these items



# Name these items



# Name these items



# Name these items



# Name these items



# Connotation

- Words have **affective** meanings
- positive connotations (*happy*)
- negative connotations (*sad*)
- positive evaluation (*great, love*)
- negative evaluation (*terrible, hate*).

# So far

- Concepts or word senses
  - Have a complex many-to-many association with words (homonymy, multiple senses)
- Have relations with each other
  - Synonymy
  - Antonymy
  - Similarity
  - Relatedness
  - Superordinate/subordinate
  - Connotation

But how to define a concept?

# Classical (“Aristotelian”) Theory of Concepts

# Classical (“Aristotelian”) Theory of Concepts

The meaning of a word:

# Classical (“Aristotelian”) Theory of Concepts

The meaning of a word:

a concept defined by **necessary** and **sufficient** conditions

# Classical (“Aristotelian”) Theory of Concepts

The meaning of a word:

a concept defined by **necessary** and **sufficient** conditions

- A **necessary** condition for being an X is a condition C that X must satisfy in order for it to be an X.
  - If not C, then not X
  - “Having four sides” is necessary to be a square.

# Classical (“Aristotelian”) Theory of Concepts

The meaning of a word:

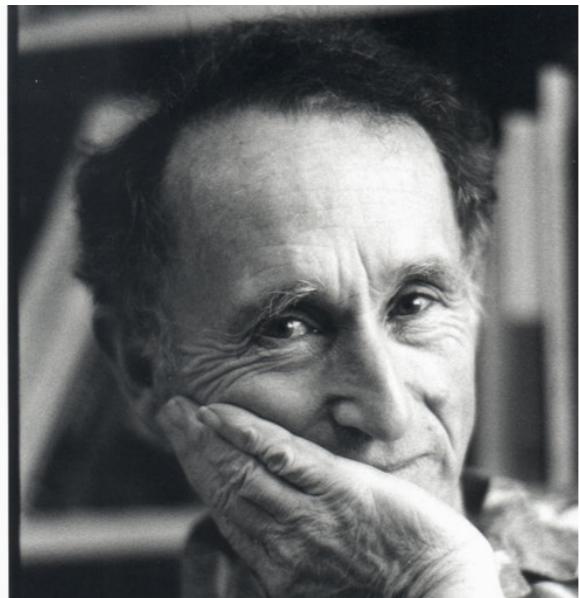
a concept defined by **necessary** and **sufficient** conditions

- A **necessary** condition for being an X is a condition C that X must satisfy in order for it to be an X.
  - If not C, then not X
  - “Having four sides” is necessary to be a square.
- A **sufficient** condition for being an X is condition such that if something satisfies condition C, then it must be an X.
  - If and only if C, then X
  - The following necessary conditions, jointly, are sufficient to be a square
    - x has (exactly) four sides
    - each of x's sides is straight
    - x is a closed figure
    - x lies in a plane
    - each of x's sides is equal in length to each of the others
    - each of x's interior angles is equal to the others (right angles)
    - the sides of x are joined at their ends

# Problem 1: The features are complex and may be context-dependent



- William Labov. 1975

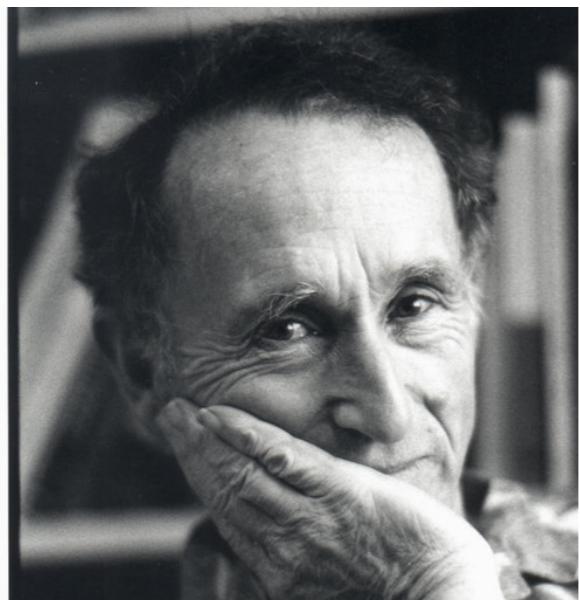


- What are these?
- Cup or bowl?

# Problem 1: The features are complex and may be context-dependent



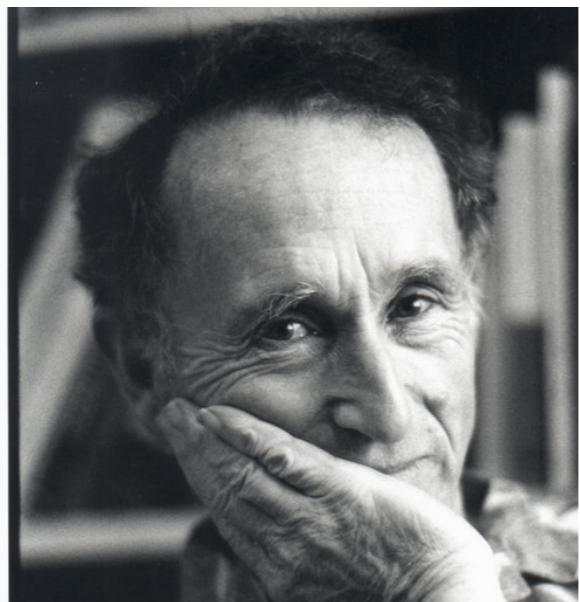
- William Labov. 1975



- What are these?
- Cup or bowl?

# Problem 1: The features are complex and may be context-dependent

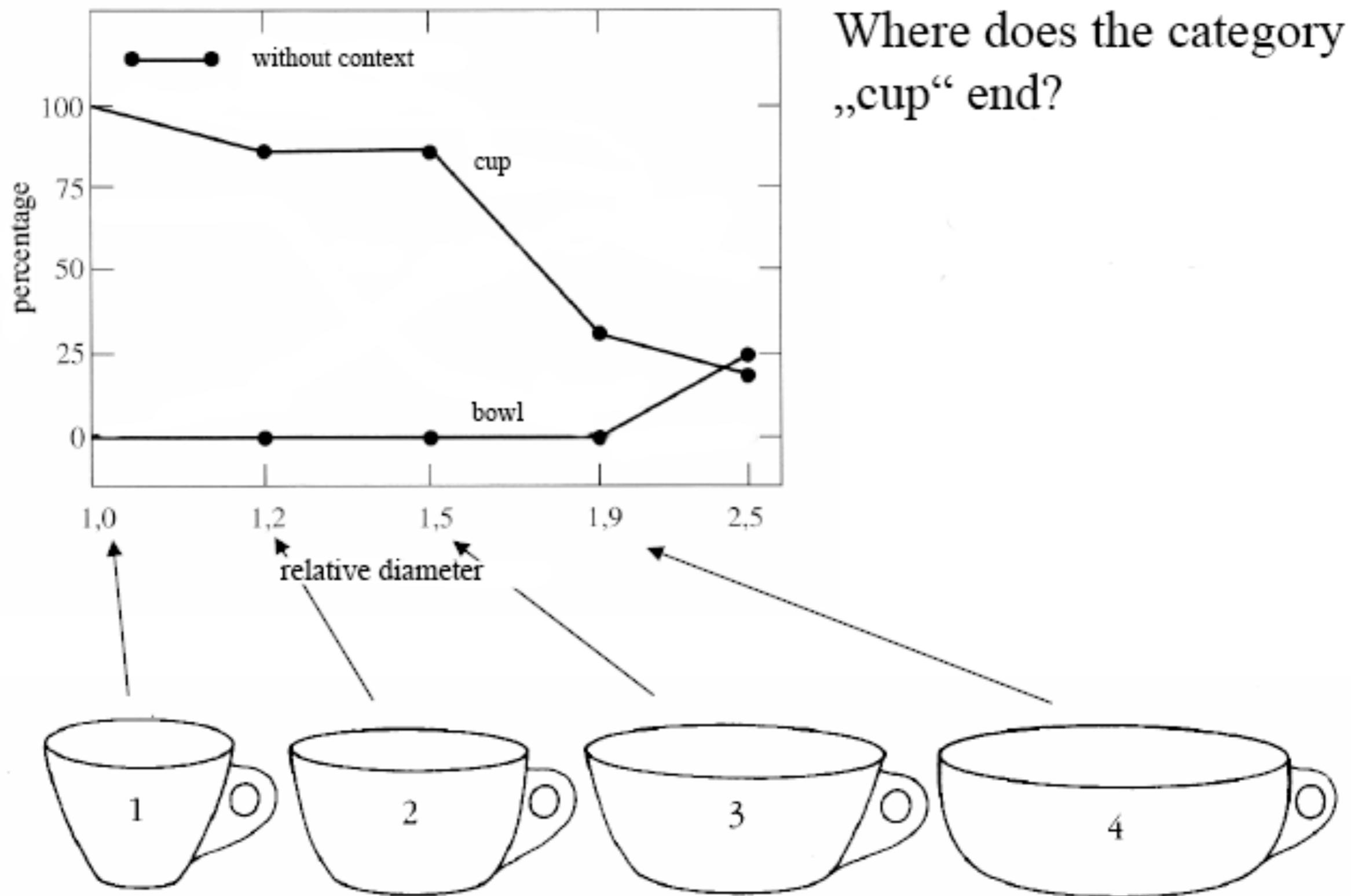
- William Labov. 1975



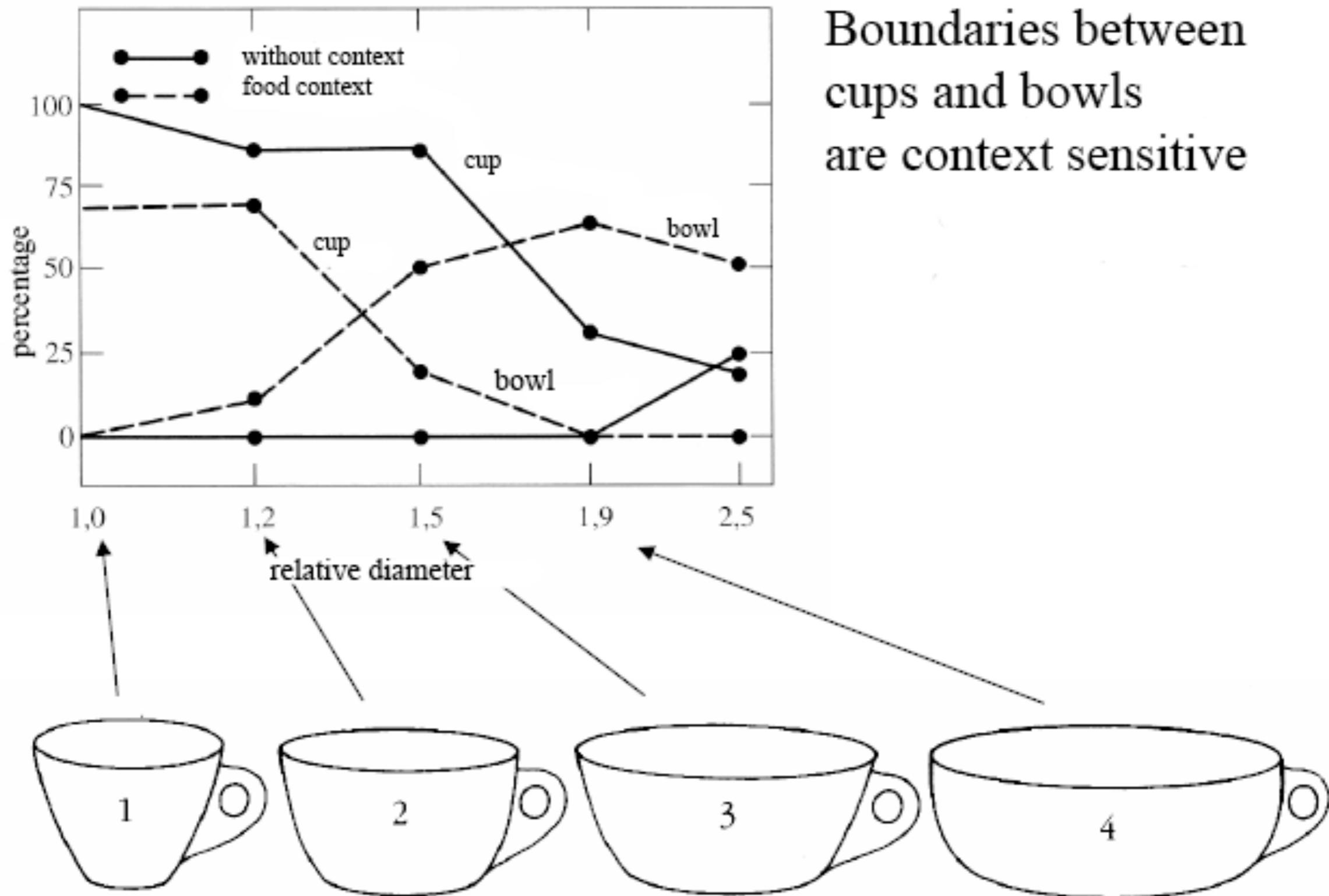
- What are these?
- Cup or bowl?



The category depends on complex features of the object (diameter, etc)



# The category depends on the context! (If there is food in it, it's a bowl)



# Labov's definition of cup

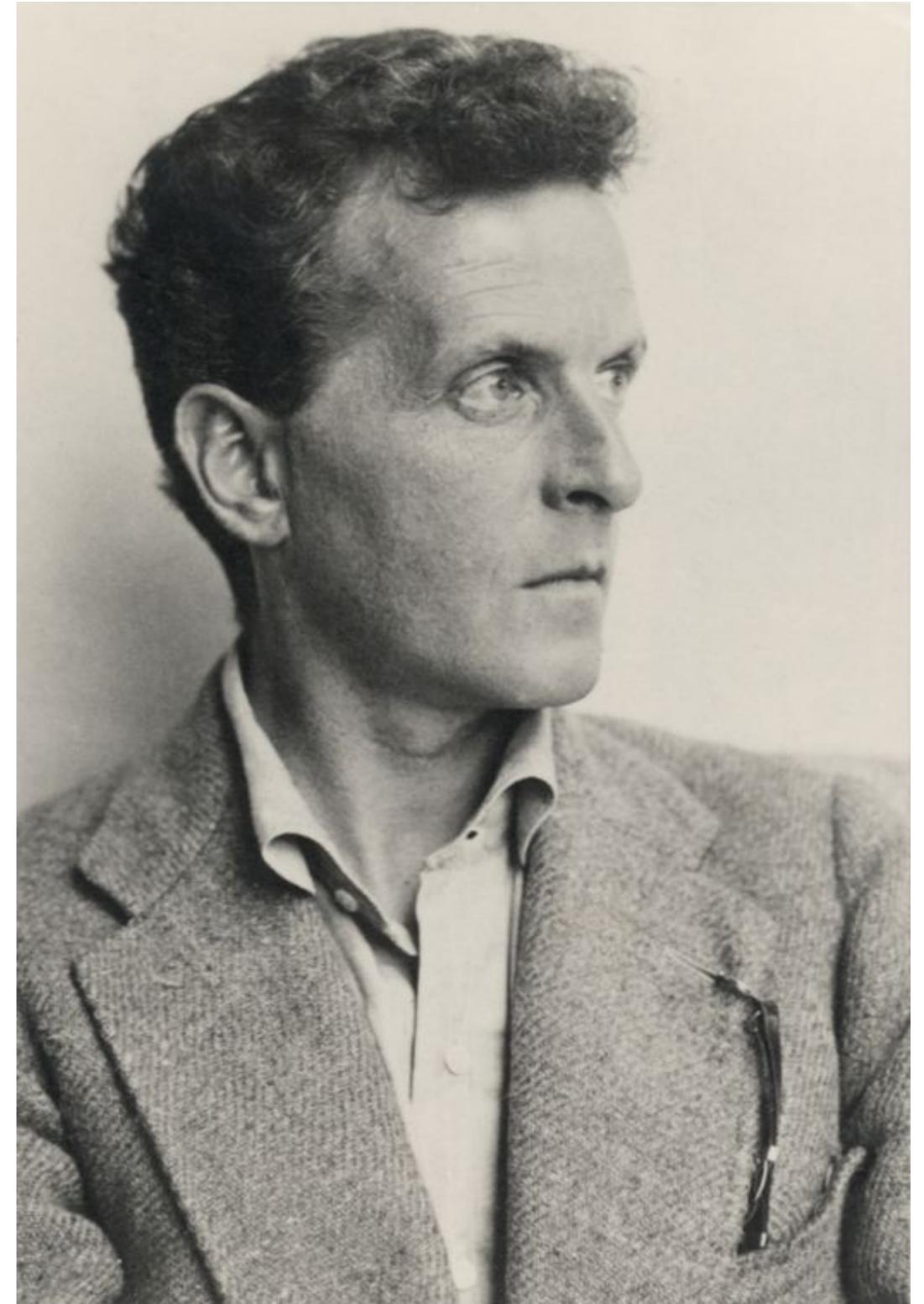
The term *cup* is used to denote round containers with a ratio of depth to width of  $1 \pm r$  where  $r \leq r_b$ , and  $r_b = \alpha_1 + \alpha_2 + \dots + \alpha_v$  and  $\alpha_i$  is a positive quality when the feature  $i$  is present and 0 otherwise.

- feature 1 = with one handle
- 2 = made of opaque vitreous material
- 3 = used for consumption of food
- 4 = used for the consumption of liquid food
- 5 = used for consumption of hot liquid food
- 6 = with a saucer
- 7 = tapering
- 8 = circular in cross-section

*Cup* is used variably to denote such containers with ratios width to depth  $1 \pm r$  where  $r_b \leq r \leq r_1$  with a probability of  $r_1 - r / r_1 - r_b$ . The quantity  $1 \pm r_b$  expresses the distance from the modal value of width to height.

# Ludwig Wittgenstein (1889-1951)

- Philosopher of language
- In his late years, a proponent of studying “ordinary language”



# Wittgenstein (1945)

## *Philosophical Investigations.*

### Paragraphs 66,67

66. Consider for example the proceedings that we call "games". I mean board-games, card-games, ball-games, Olympic games, and so on. What is common to them all?—Don't say: "There *must* be something common, or they would not be called 'games'"—but *look and see* whether there is anything common to all.—For if you look at them you will not see something that is common to *all*, but similarities, relationships, and a whole series of them at that. To repeat: don't think, but look!—Look for example at board-games, with their multifarious relationships. Now pass to card-games; here you find many correspondences with the first group, but many common features drop out, and others appear. When we pass next to ball-games, much that is common is retained, but much is lost.—Are they all 'amusing'? Compare chess with noughts and crosses. Or is there always winning and losing, or competition between players? Think of patience. In ball games there is winning and losing; but when a child throws his ball at the wall and catches it again, this feature has disappeared. Look at the parts played by skill and luck; and at the difference between skill in chess and skill in tennis. Think now of games like ring-a-ring-a-roses; here is the element of amusement, but how many other characteristic features have disappeared! And we can go through the many, many other groups of games in the same way; can see how similarities crop up and disappear.

And the result of this examination is: we see a complicated network of similarities overlapping and criss-crossing: sometimes overall similarities, sometimes similarities of detail.

67. I can think of no better expression to characterize these similarities than "family resemblances"; for the various resemblances between members of a family: build, features, colour of eyes, gait, temperament, etc. etc. overlap and criss-cross in the same way.—And I shall say: 'games' form a family.

And for instance the kinds of number form a family in the same way. Why do we call something a "number"? Well, perhaps because it has a—direct—relationship with several things that have hitherto been called number; and this can be said to give it an indirect relationship to other things we call the same name. And we extend our concept of number as in spinning a thread we twist fibre on fibre. And the strength of the thread does not reside in the fact that some one fibre runs through its whole length, but in the overlapping of many fibres.

But if someone wished to say: "There is something common to all these constructions—namely the disjunction of all their common properties"—I should reply: Now you are only playing with words. One might as well say: "Something runs through the whole thread—namely the continuous overlapping of those fibres".

# What is a game?

# Wittgenstein's thought experiment on "What is a game":

PI #66:

"Don't say "there must be something common, or they would not be called 'games'" — but *look and see* whether there is anything common to all"

Is it amusing?

Is there competition?

Is there long-term strategy?

Is skill required?

Must luck play a role?

Are there cards?

Is there a ball?

# Family Resemblance

Game 1	Game 2	Game 3	Game 4
ABC	BCD	ACD	ABD

“each item has at least one, and probably several, elements in common with one or more items, but no, or few, elements are common to all items” – Rosch and Mervis

How about a radically different  
approach?

# Ludwig Wittgenstein

PI #43:

"The meaning of a word is its use in  
the language"

# Let's define words by their usages

- In particular, words are defined by their environments (the words around them)
- Zellig Harris (1954): If A and B have almost identical environments we say that they are synonyms.

# What does ongchoi mean?

# What does ongchoi mean?

- Suppose you see these sentences:
  - Ong choi is delicious **sautéed with garlic**.
  - Ong choi is superb **over rice**
  - Ong choi **leaves** with salty sauces

# What does ongchoi mean?

- Suppose you see these sentences:
  - Ong choi is delicious **sautéed with garlic**.
  - Ong choi is superb **over rice**
  - Ong choi **leaves** with salty sauces
- And you've also seen these:
  - ...spinach **sautéed with garlic over rice**
  - Chard stems and **leaves** are **delicious**
  - Collard greens and other **salty** leafy greens

# What does ongchoi mean?

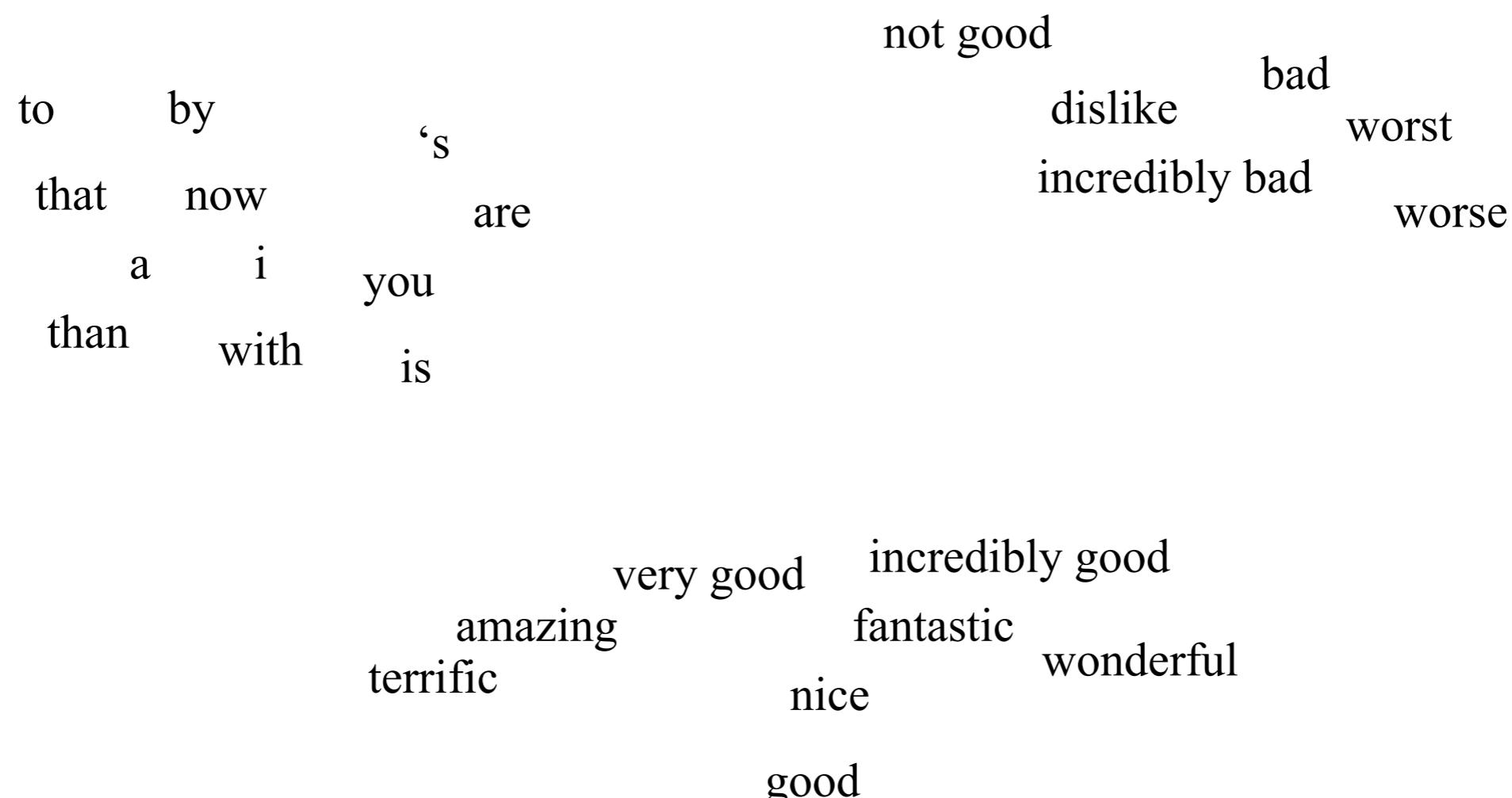
- Suppose you see these sentences:
  - Ong choi is delicious **sautéed with garlic**.
  - Ong choi is superb **over rice**
  - Ong choi **leaves** with salty sauces
- And you've also seen these:
  - ...spinach **sautéed with garlic over rice**
  - Chard stems and **leaves** are **delicious**
  - Collard greens and other **salty** leafy greens
- Conclusion:
  - Ongchoi is a leafy green like spinach, chard, or collard greens

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



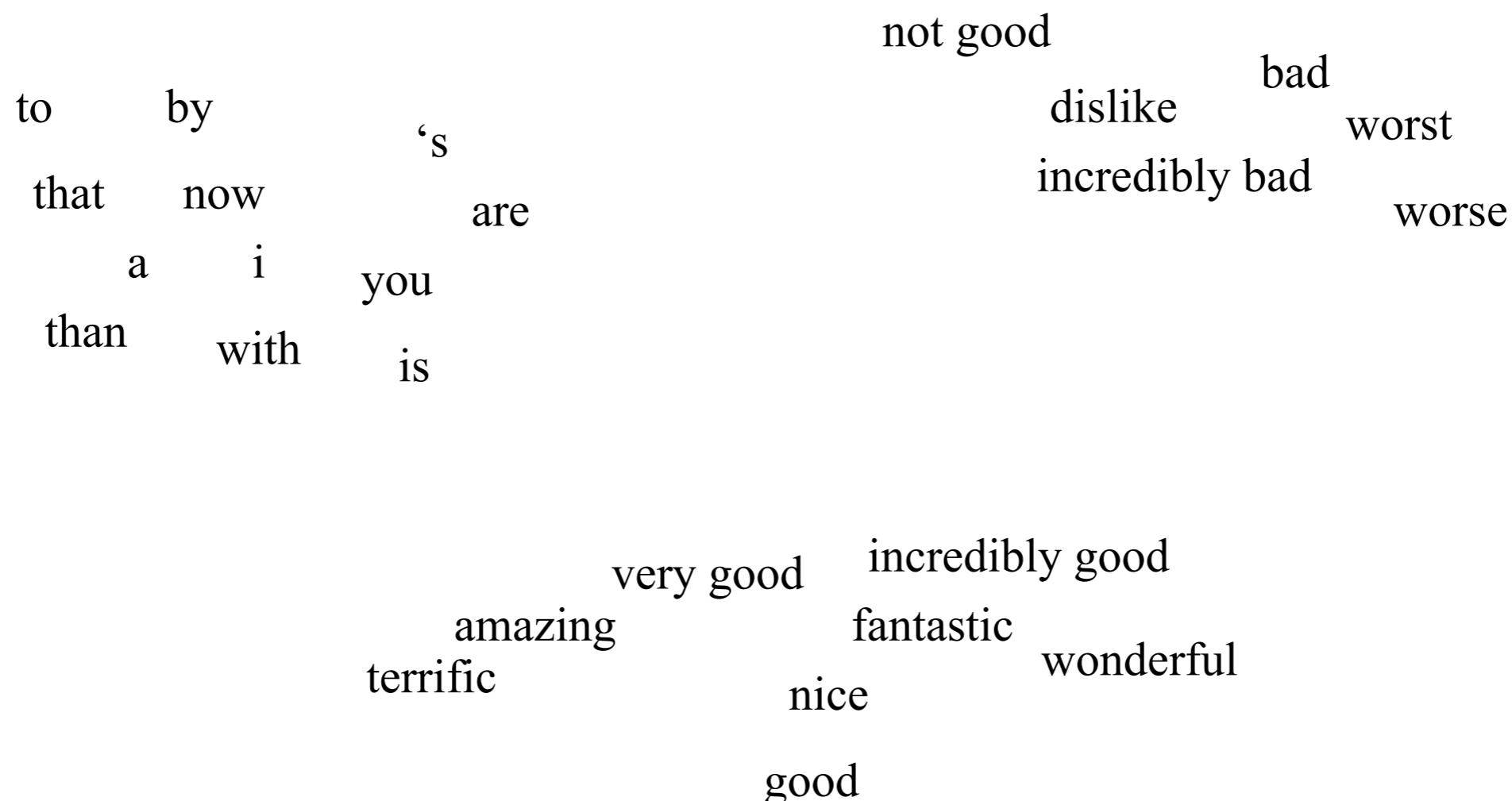
Yamaguchi, Wikimedia Commons, public domain

# We'll build a new model of meaning focusing on similarity



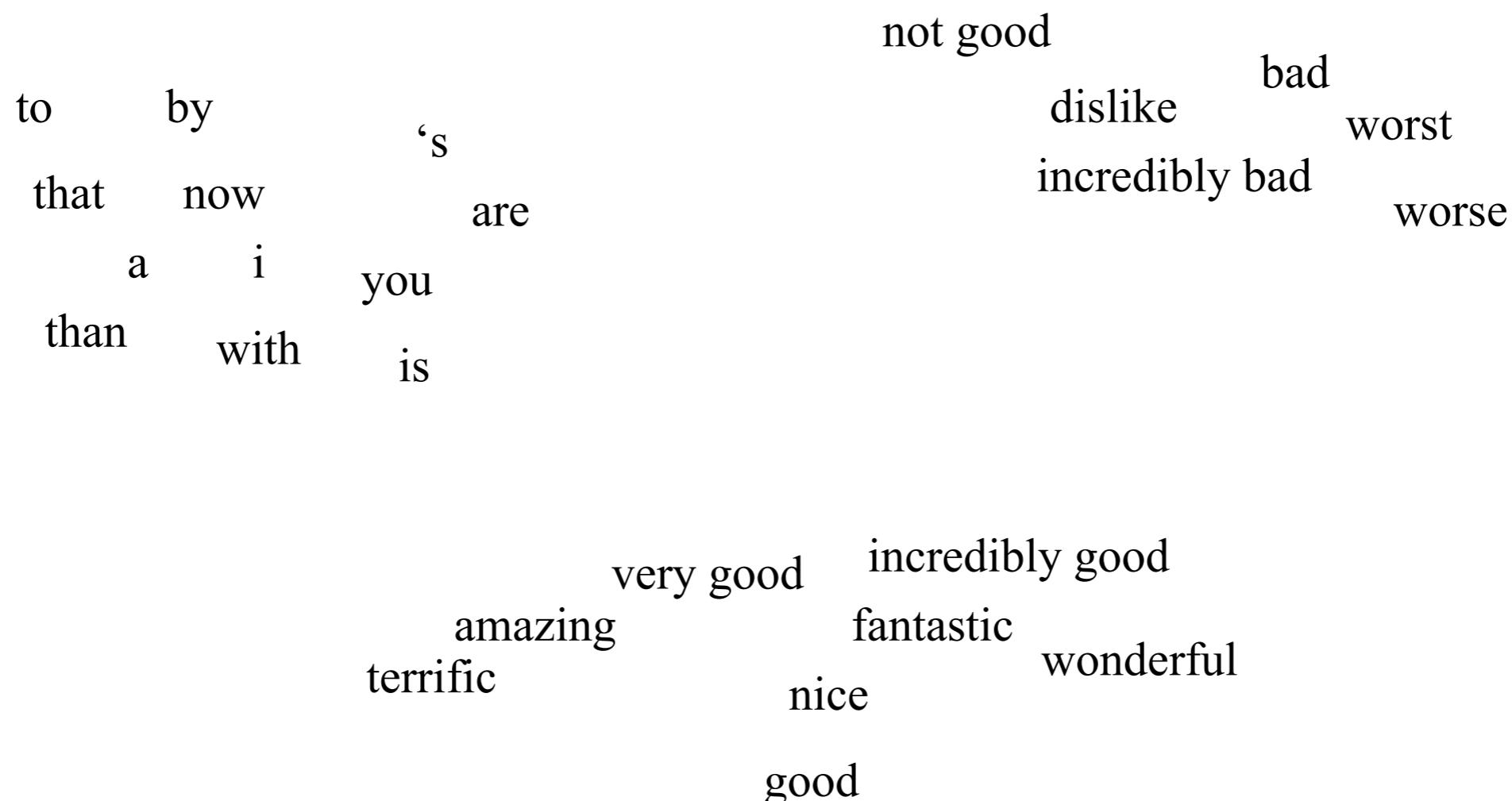
# We'll build a new model of meaning focusing on similarity

- Each word = a **vector**



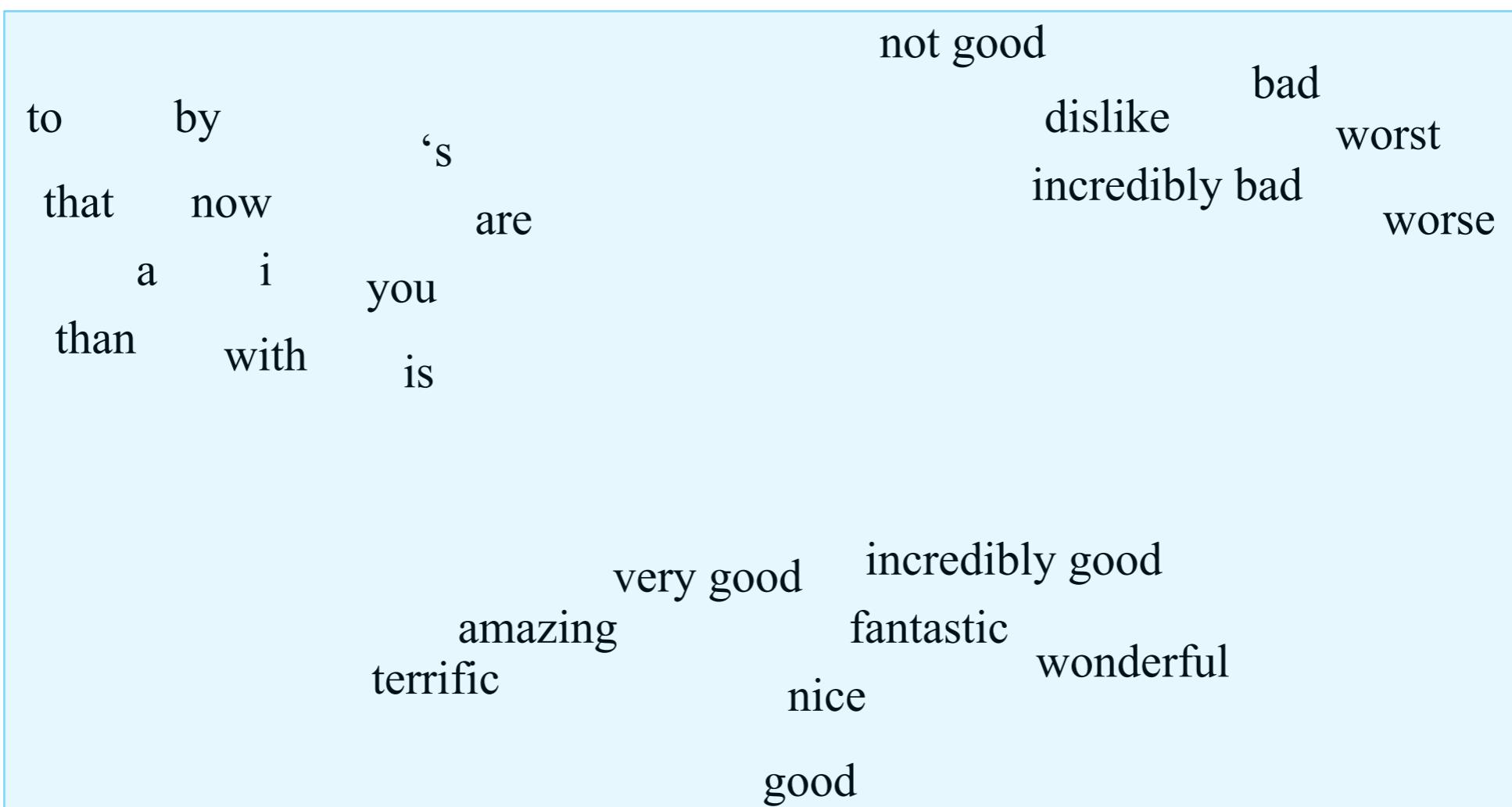
# We'll build a new model of meaning focusing on similarity

- Each word = a **vector**
  - Not just "word" or word45.



# We'll build a new model of meaning focusing on similarity

- Each word = a **vector**
  - Not just "word" or word45.
  - Similar words are "nearby in space"



We define a word as a vector

# We define a word as a vector

- Called an "**embedding**" because it's embedded into a space (you might also see this called a **projection**)

# We define a word as a vector

- Called an "**embedding**" because it's embedded into a space (you might also see this called a **projection**)
- The standard way to represent meaning in NLP

# We define a word as a vector

- Called an "**embedding**" because it's embedded into a space (you might also see this called a **projection**)
- The standard way to represent meaning in NLP
- Fine-grained model of meaning for similarity
  - NLP tasks like sentiment analysis
    - With words, requires **same** word to be in training and test
    - With embeddings: ok if **similar** words occurred!!!
  - Question answering, conversational agents, etc



# Old school vectors

# Lexical semantics

“You shall know a word by the company it keeps”

[Firth 1957]



everyone likes \_\_\_\_\_

everyone likes \_\_\_\_\_

a bottle of \_\_\_\_\_ is on the table

everyone likes \_\_\_\_\_

a bottle of \_\_\_\_\_ is on the table

\_\_\_\_\_ makes you drunk

everyone likes \_\_\_\_\_

a bottle of \_\_\_\_\_ is on the table

\_\_\_\_\_ makes you drunk

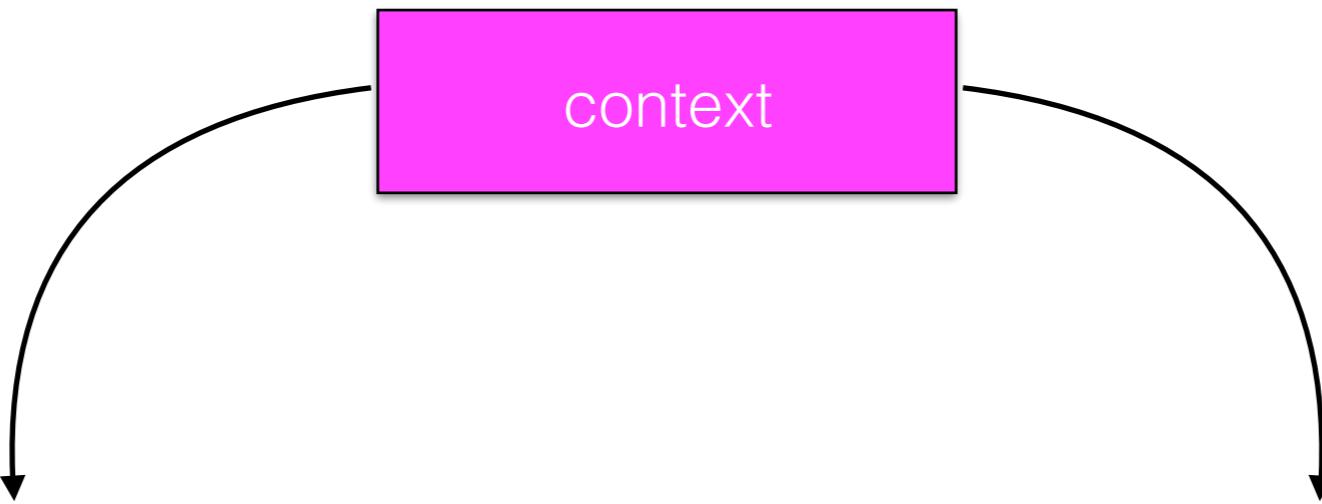
a cocktail with \_\_\_\_\_ and seltzer

# Context

“You shall know a word by the company it keeps”

[Firth 1957]

- A few different ways we can encode the notion of “company” (or **context**).



everyone likes \_\_\_\_\_

a bottle of \_\_\_\_\_ is on the table

\_\_\_\_\_ makes you drunk

a cocktail with \_\_\_\_\_ and seltzer

# Distributed representation

- Vector representation that encodes information about the **distribution** of contexts a word appears in
- Words that appear in similar contexts have similar representations (and similar meanings, by the **distributional hypothesis**).

# Term-document matrix

	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear
knife	1	1	4	2		2		2
dog	2		6	6		2		12
sword	17	2	7	12		2		17
love	64		135	63		12		48
like	75	38	34	36	34	41	27	44

Context = appearing in the same document.

# Vector

Vector  
representation of  
the **document**;  
vector size = V

Hamlet
1
2
17
64
75

King Lear
2
12
17
48
44

# Vectors

knife	1	1	4	2		2		2
-------	---	---	---	---	--	---	--	---

sword	17	2	7	12		2		17
-------	----	---	---	----	--	---	--	----

Vector representation of the  
**term**; vector size = number  
of documents

# Weighting dimensions

- Not all dimensions are equally informative

# TF-IDF

- Term frequency-inverse document frequency
- A scaling to represent a feature as function of how frequently it appears in a data point **but accounting for its frequency in the overall collection**
- IDF for a given term = the number of documents in collection / number of documents that contain term

# TF-IDF

- Term frequency ( $tf_{t,d}$ ) = the number of times term t occurs in document d
- Inverse document frequency = inverse fraction of number of documents containing ( $D_t$ ) among total number of documents N

$$tfidf(t, d) = tf_{t,d} \times \log \frac{N}{D_t}$$

# IDF

	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear	IDF
knife	1	1	4	2		2		2	0.12
dog	2		6	6		2		12	0.20
sword	17	2	7	12		2		17	0.12
love	64		135	63		12		48	0.20
like	75	38	34	36	34	41	27	44	0

IDF for the informativeness of the terms when comparing documents

# PMI

- Mutual information provides a measure of how independent two **variables** (X and Y) are.
- Pointwise mutual information measures the independence of two **outcomes** (x and y)

# PMI

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

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

# PMI

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

w = word, c = context

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

# PMI

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

w = word, c = context

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

What's this value for w and c  
that never occur together?

# PMI

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

w = word, c = context

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

What's this value for w and c  
that never occur together?

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

# Weighting PMI

# Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values

$$PPMI(w_1, w_2) = \max(0, \log_2 \frac{P(w_1, w_2)}{p(w_1)p(w_2)})$$

# Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values

$$PPMI(w_1, w_2) = \max(0, \log_2 \frac{P(w_1, w_2)}{p(w_1)p(w_2)})$$

- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)

# Weighting PMI: Giving rare context words slightly higher probability

# Weighting PMI: Giving rare context words slightly higher probability

- Raise the context word ( $w_2$ ) probabilities to  $\alpha = 0.75$  :

$$PPMI(w_1, w_2) = \max(0, \log_2 \frac{P(w_1, w_2)}{p(w_1)p_a(w_2)})$$

$$p_a(w_i) = \frac{\text{count}(w_i)^\alpha}{\sum_{w_j} \text{count}(w_j)^\alpha}$$

# Weighting PMI: Giving rare context words slightly higher probability

- Raise the context word ( $w_2$ ) probabilities to  $\alpha = 0.75$  :

$$PPMI(w_1, w_2) = \max(0, \log_2 \frac{P(w_1, w_2)}{p(w_1)p_a(w_2)})$$

$$p_a(w_i) = \frac{\text{count}(w_i)^\alpha}{\sum_{w_j} \text{count}(w_j)^\alpha}$$

- This helps because  $p_a(c) > p(c)$  for rare  $c$

# Weighting PMI: Giving rare context words slightly higher probability

- Raise the context word ( $w_2$ ) probabilities to  $\alpha = 0.75$  :

$$PPMI(w_1, w_2) = \max(0, \log_2 \frac{P(w_1, w_2)}{p(w_1)p_a(w_2)})$$

$$p_a(w_i) = \frac{\text{count}(w_i)^\alpha}{\sum_{w_j} \text{count}(w_j)^\alpha}$$

- This helps because  $p_a(c) > p(c)$  for rare  $c$
- Consider two events,  $P(a) = .99$  and  $P(b) = .01$

$$p_a(a) = \frac{.99^{0.75}}{.99^{0.75} + .01^{0.75}} = 0.97 \quad p_a(b) = \frac{.01^{0.75}}{.99^{0.75} + .01^{0.75}} = 0.03$$

	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear	total
knife	1	1	4	2		2		2	12
dog	2		6	6		2		12	28
sword	17	2	7	12		2		17	57
love	64		135	63		12		48	322
like	75	38	34	36	34	41	27	44	329
total	159	41	186	119	34	59	27	123	748

$$PMI(\text{love}, \text{R&J}) = \frac{\frac{135}{748}}{\frac{186}{748} \times \frac{322}{748}}$$

# Term-term matrix

- Rows and columns are both words; cell counts = the number of times word  $w_i$  and  $w_j$  show up in the **same document**.
- More common to define document = some smaller context (e.g., a window of 5 tokens)

# Term-document matrix

	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear
knife	1	1	4	2		2		2
dog	2		6	6		2		12
sword	17	2	7	12		2		17
love	64		135	63		12		48
like	75	38	34	36	34	41	27	44

# Term-term matrix

	knife	dog	sword	love	like
knife	6	5	6	5	5
dog	5	5	5	5	5
sword	6	5	6	5	5
love	5	5	5	5	5
like	5	5	5	5	8

*write a book*  
*write a poem*

- First-order co-occurrence (syntagmatic association): **write** co-occurs with **book** in the same sentence.
- Second-order co-occurrence (paradigmatic association): **book** co-occurs with **poem** (since each co-occur with **write**)

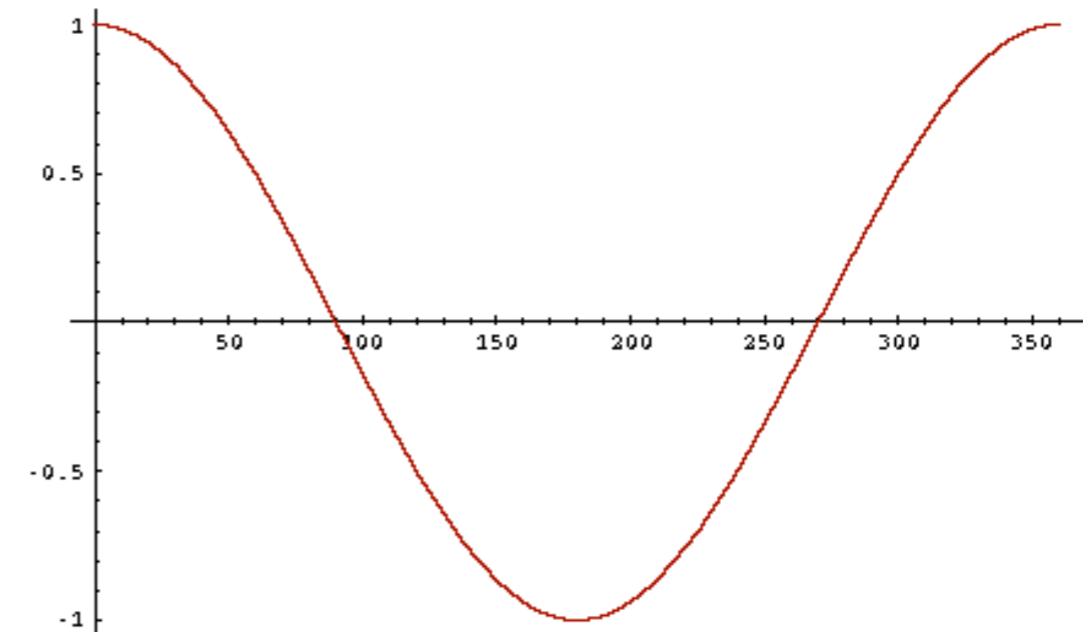
# Cosine Similarity

$$\cos(x, y) = \frac{\sum_{i=1}^F x_i y_i}{\sqrt{\sum_{i=1}^F x_i^2} \sqrt{\sum_{i=1}^F y_i^2}}$$

- We can calculate the cosine similarity of two vectors to judge the degree of their similarity [Salton 1971]
- Euclidean distance measures the **magnitude** of distance between two points
- Cosine similarity measures their **orientation**

# Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal



- Frequency is non-negative, so cosine range 0-1

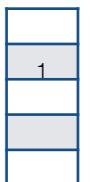


# Quick Recap

# Feed-Forward Neural LM

Simple feed-forward multilayer perceptron  
(e.g., one hidden layer)

input  $x$  = vector concatenation of a conditioning context of  
fixed size  $k$



$$x = [v(w_1); \dots; v(w_k)]$$

Bengio et al. 2003

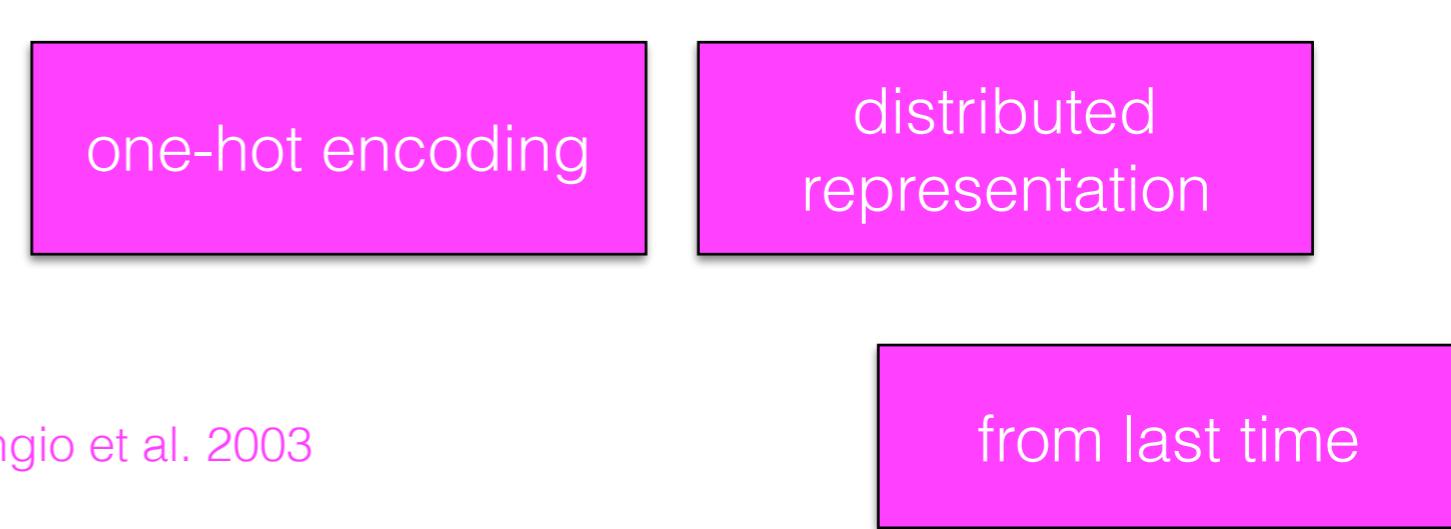
from last time

$$x = [v(w_1); \dots; v(w_k)]$$

$w_1$  = tried  
 $w_2$  = to  
 $w_3$  = prepare  
 $w_4$  = residents

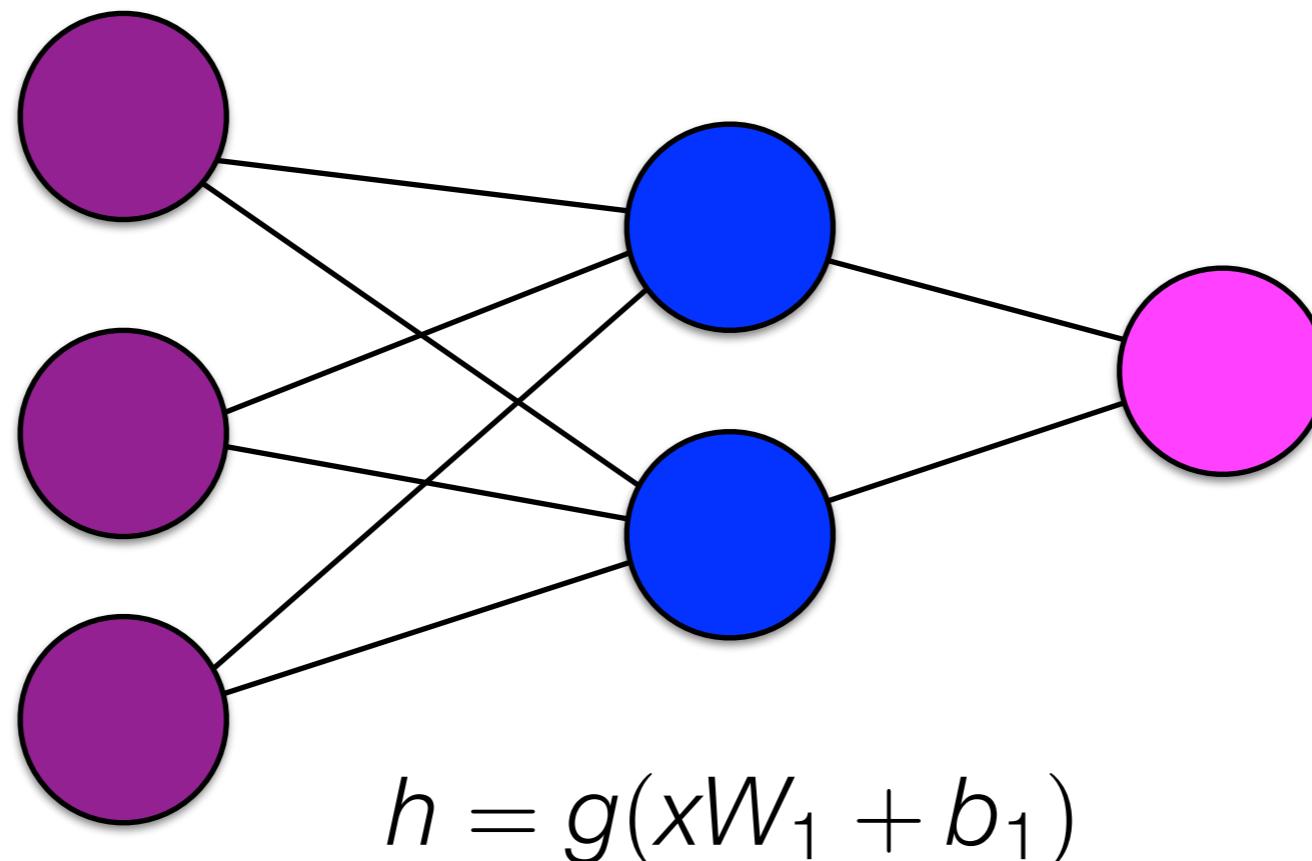


Bengio et al. 2003



$$W_1 \in \mathbb{R}^{kD \times H}$$
$$b_1 \in \mathbb{R}^H$$

$$W_2 \in \mathbb{R}^{H \times V}$$
$$b_2 \in \mathbb{R}^V$$



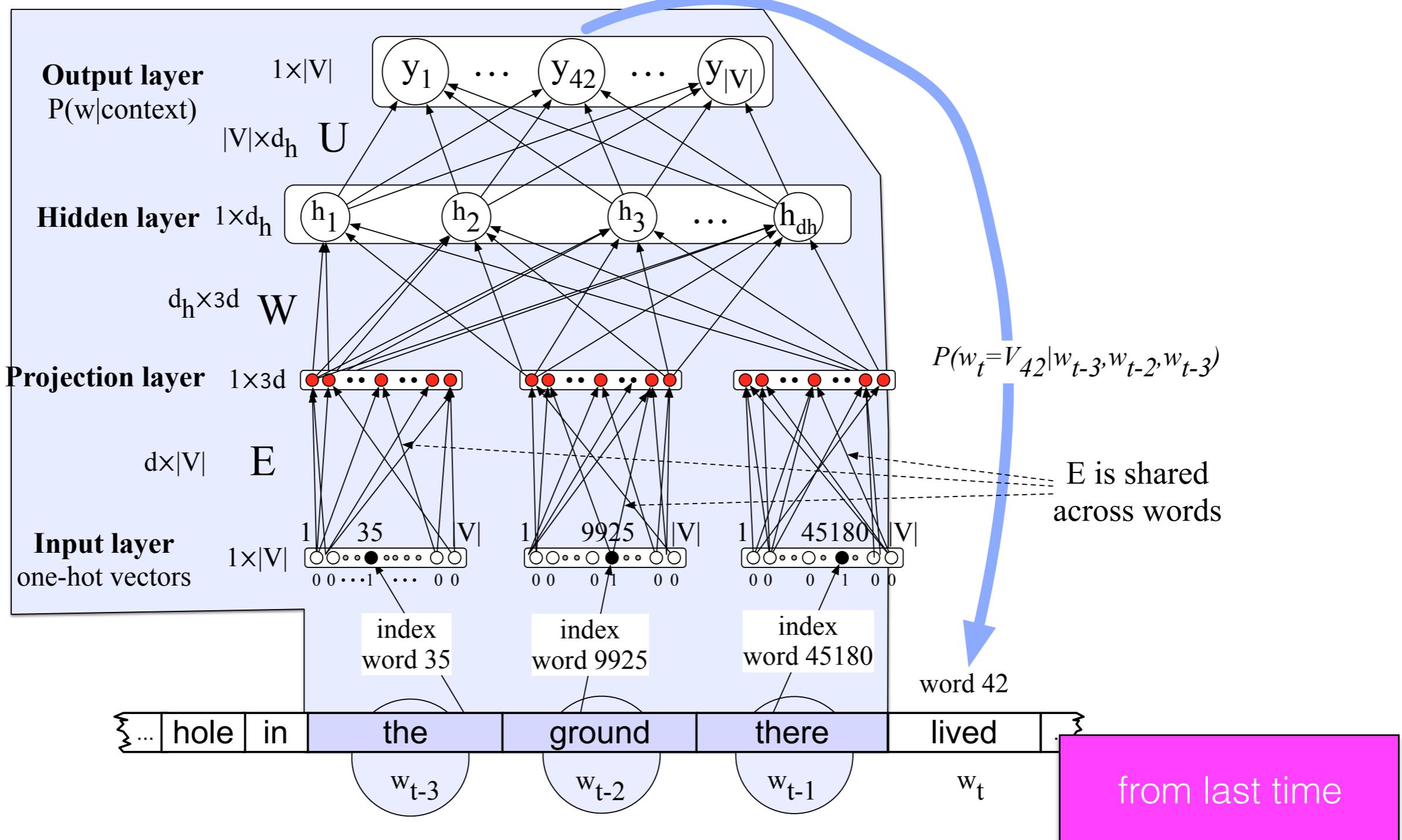
$$x = [v(w_1); \dots; v(w_k)]$$

$$\hat{y} = \text{softmax}(hW_2 + b_2)$$

Bengio et al. 2003

from last time

# Another look at the Feed Forward Network



# Softmax

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

from last time

# Softmax

$$P(Y = y \mid X = x; \beta) = \frac{\exp(x^\top \beta_y)}{\sum_{y' \in \mathcal{Y}} \exp(x^\top \beta_{y'})}$$

What's the equivalent of softmax if we only have two classes?

from last time

# Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

from last time

# Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

from last time

# Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

from last time

# Neural LM

conditioning context

tried to prepare residents for the hardships of recovery from the

y

from last time

# Neural LM

conditioning context

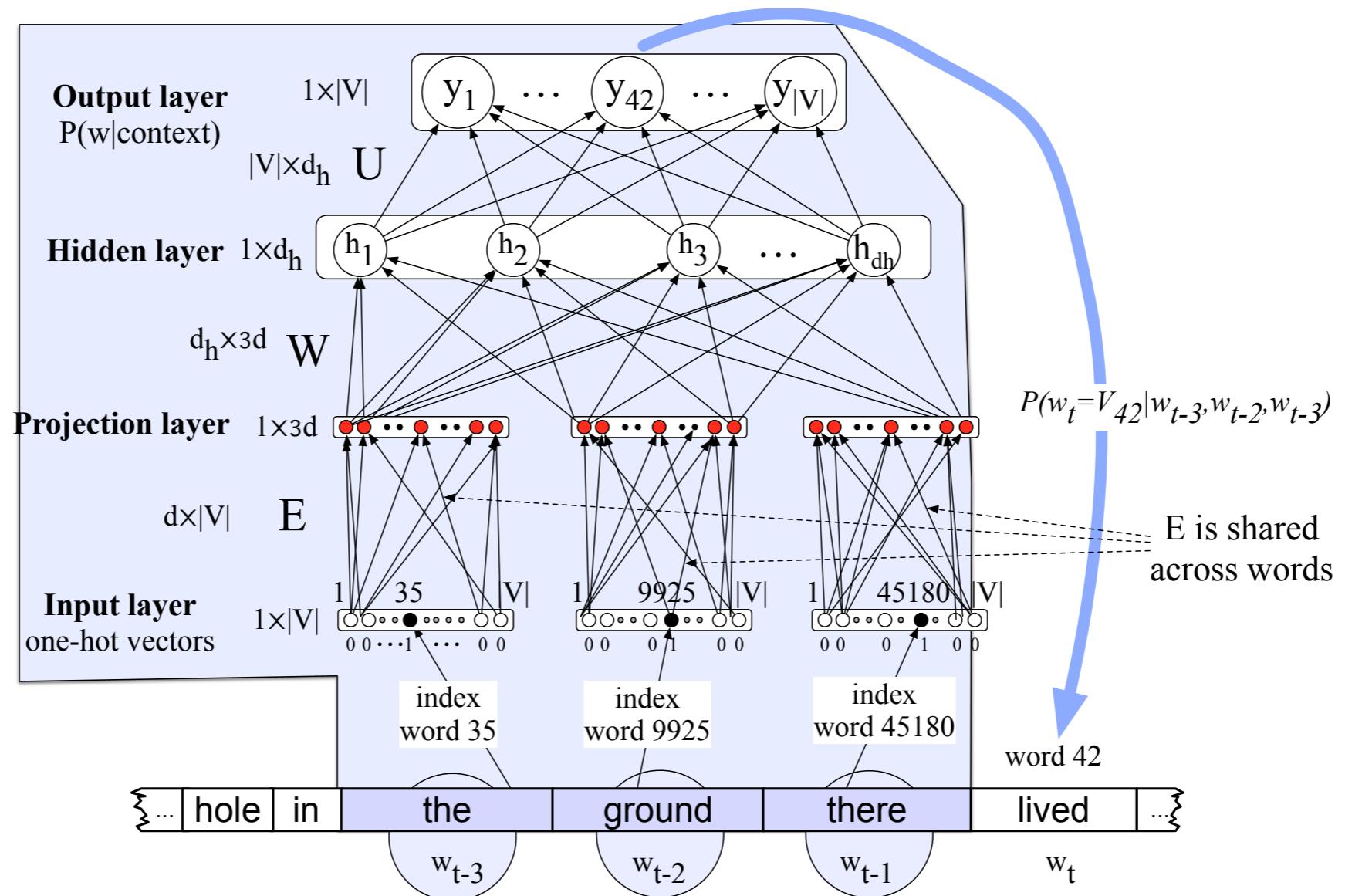
tried to prepare residents for the hardships of recovery from the

y

from last time

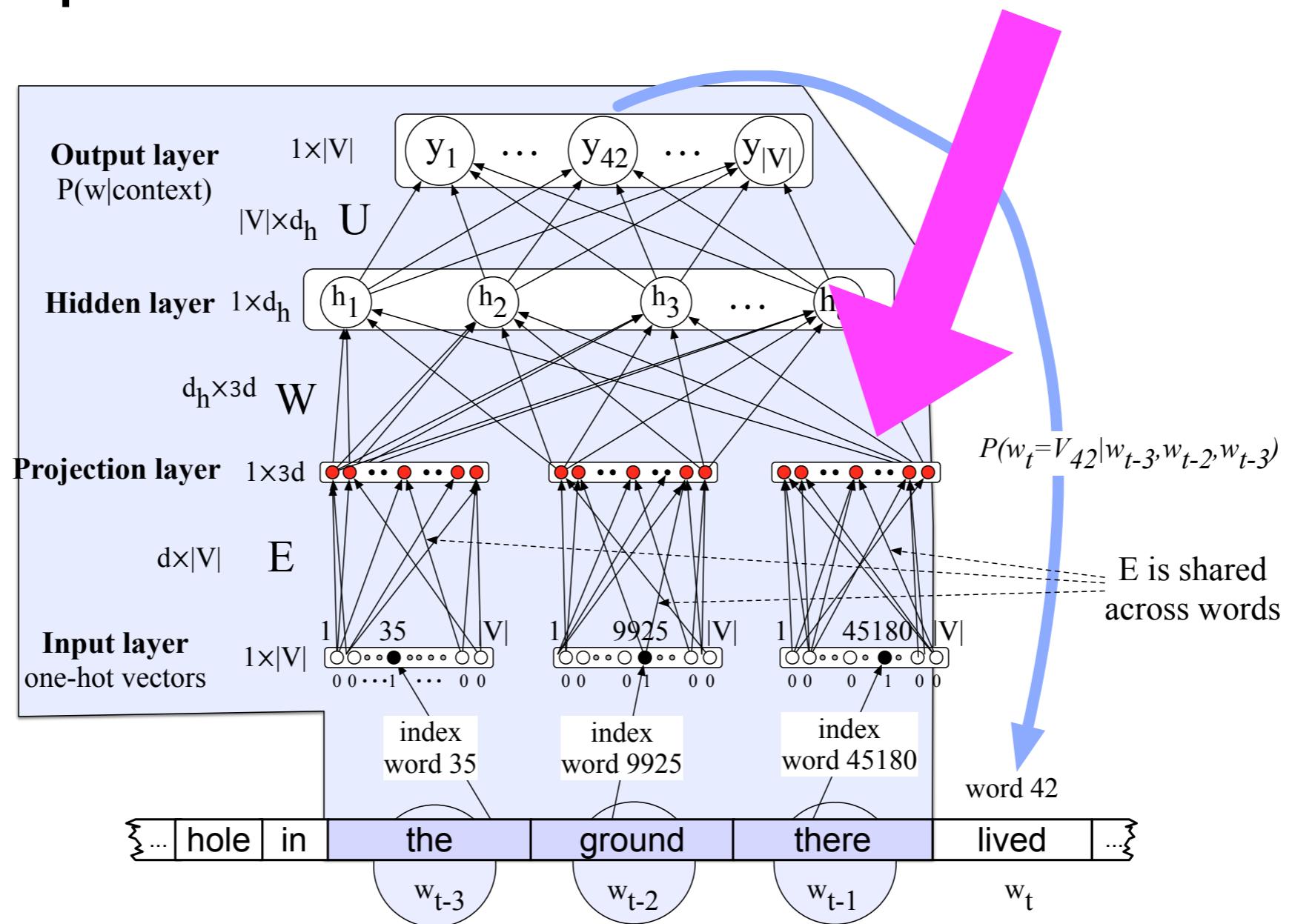
# Today's question:

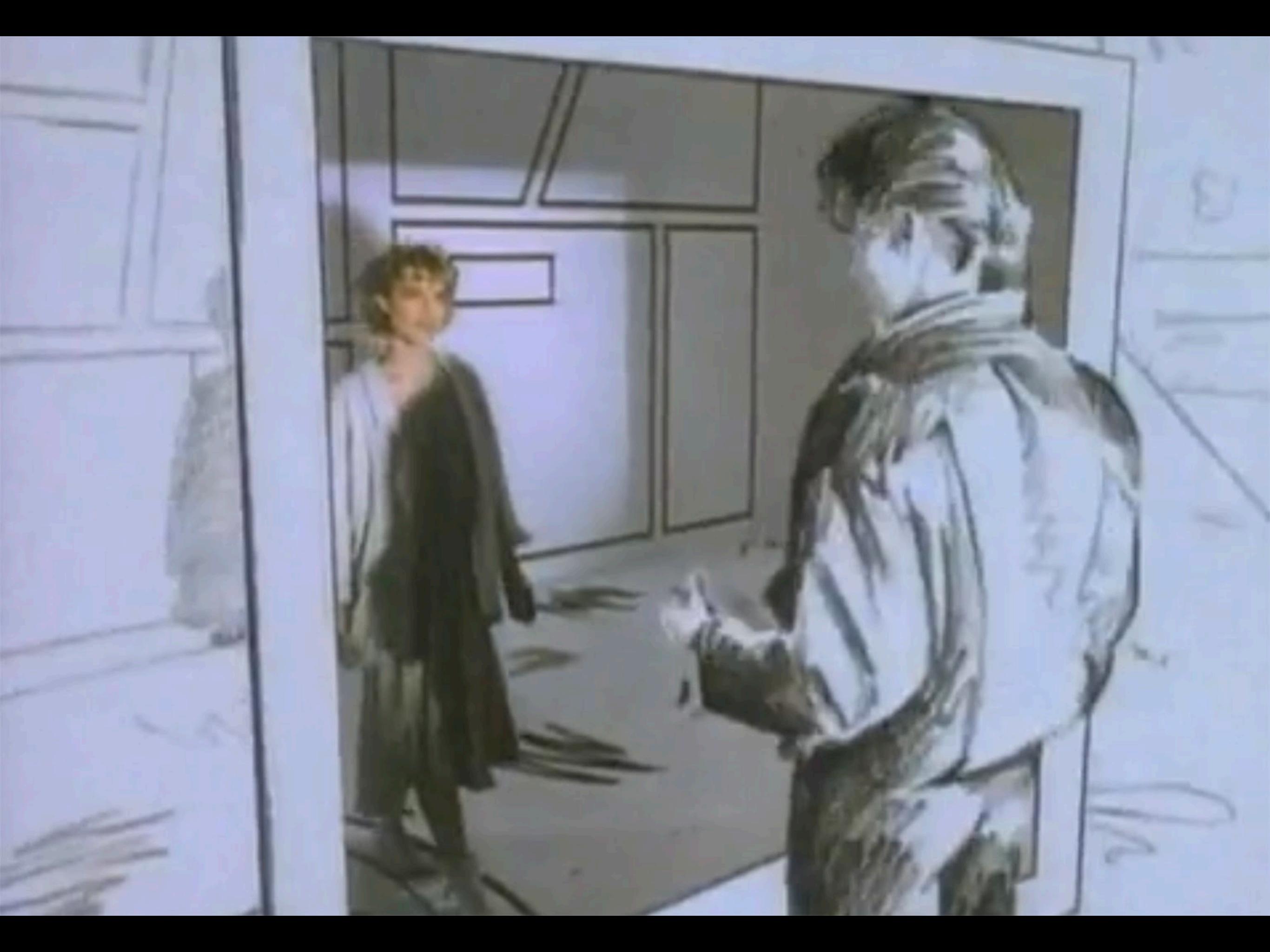
## Where do we get those distributed representations of words?



# Today's question:

## Where do we get those distributed representations of words?

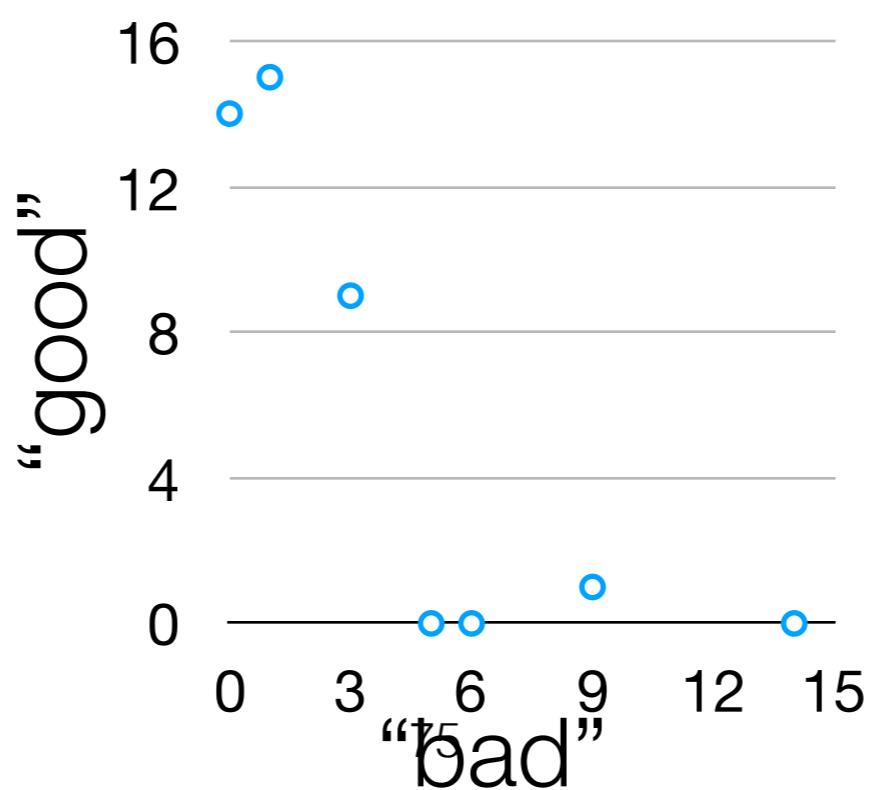




# Dimensionality Reduction

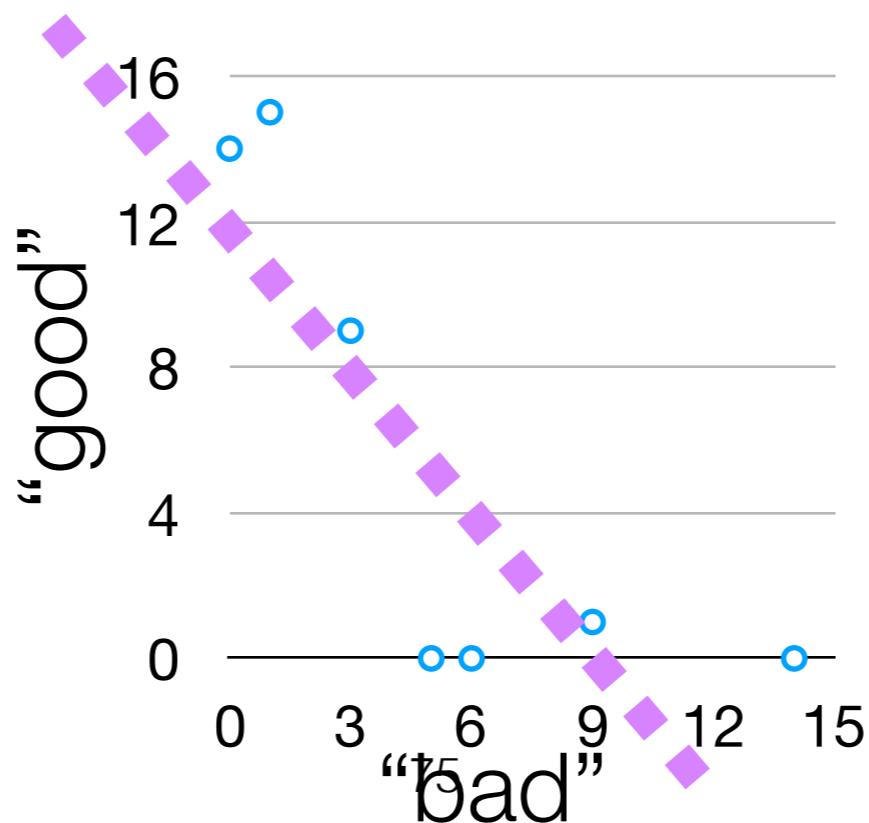
	good	bad
action	3	9
vampires	0	14
noir	6	0
oscar	5	0
drama	9	1
dumbledore	14	0
romcom	1	15

	good	bad
action	3	9
vampires	0	14
noir	6	0
oscar	5	0
drama	9	1
dumbledore	14	0
romcom	1	15



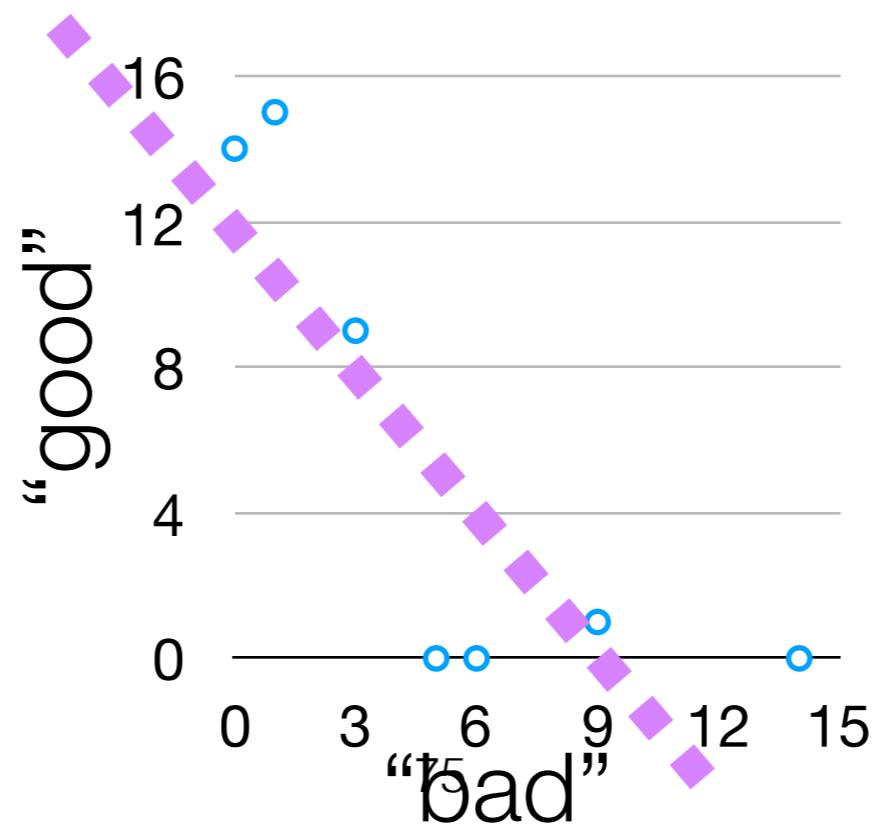
	good	bad
action	3	9
vampires	0	14
noir	6	0
oscar	5	0
drama	9	1
dumbledore	14	0
romcom	1	15

With two dimensions, fitting a line gives us a way of representing each word with one value along the time —i.e., **one dimension** instead of two



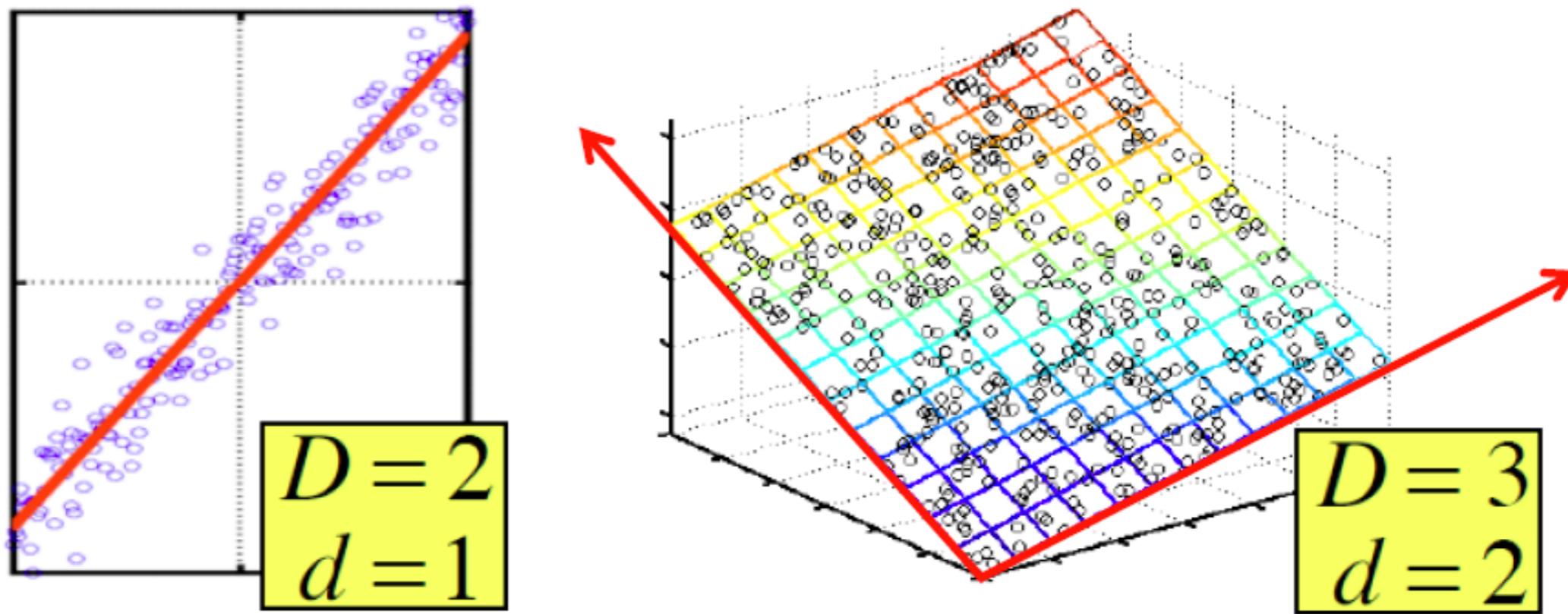
	good	bad	great	awesome	hate
action	3	9	1	1	14
vampires	0	14	0	1	19
noir	6	0	3	1	0
oscar	5	0	2	4	0
drama	9	1	2	1	3
dumbledore	14	0	9	8	2
romcom	1	15	1	1	11

With two dimensions, fitting a line gives us a way of representing each word with one value along the time —i.e., **one dimension** instead of two

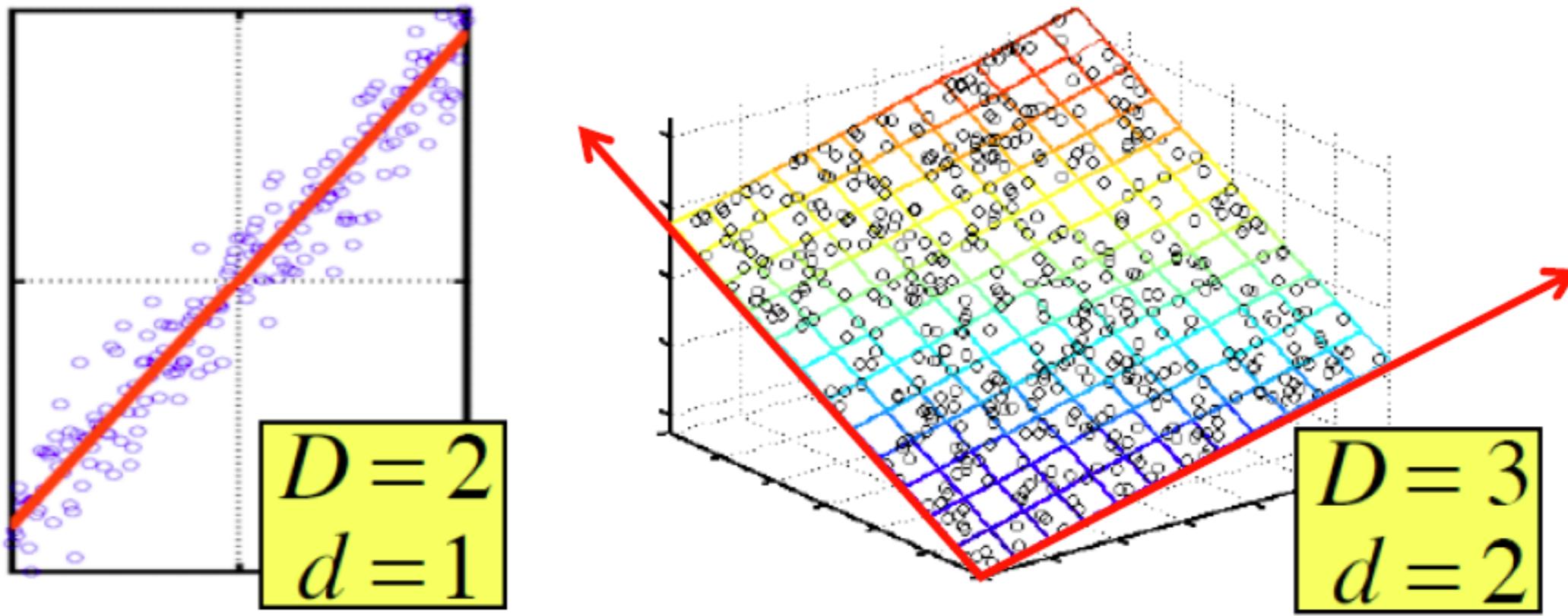


How do we reduce dimensionality when we have **multiple dimensions**?

# Dimensionality Reduction

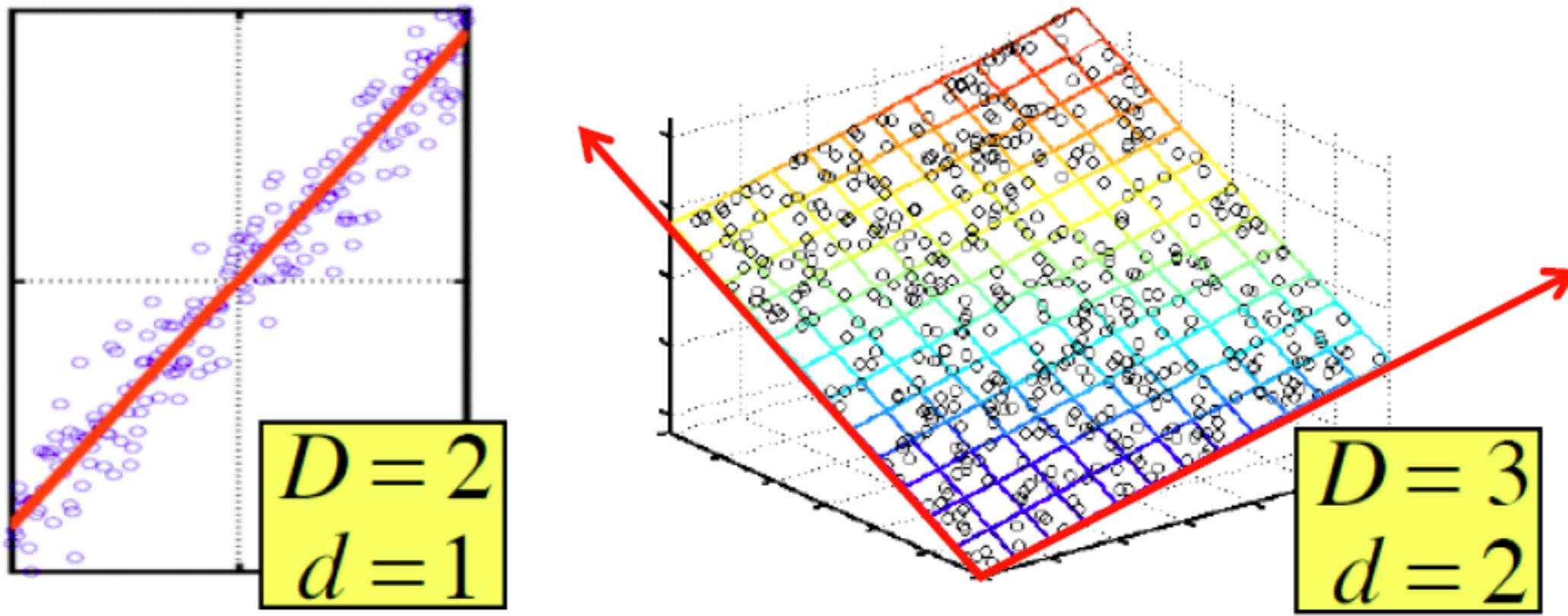


# Dimensionality Reduction



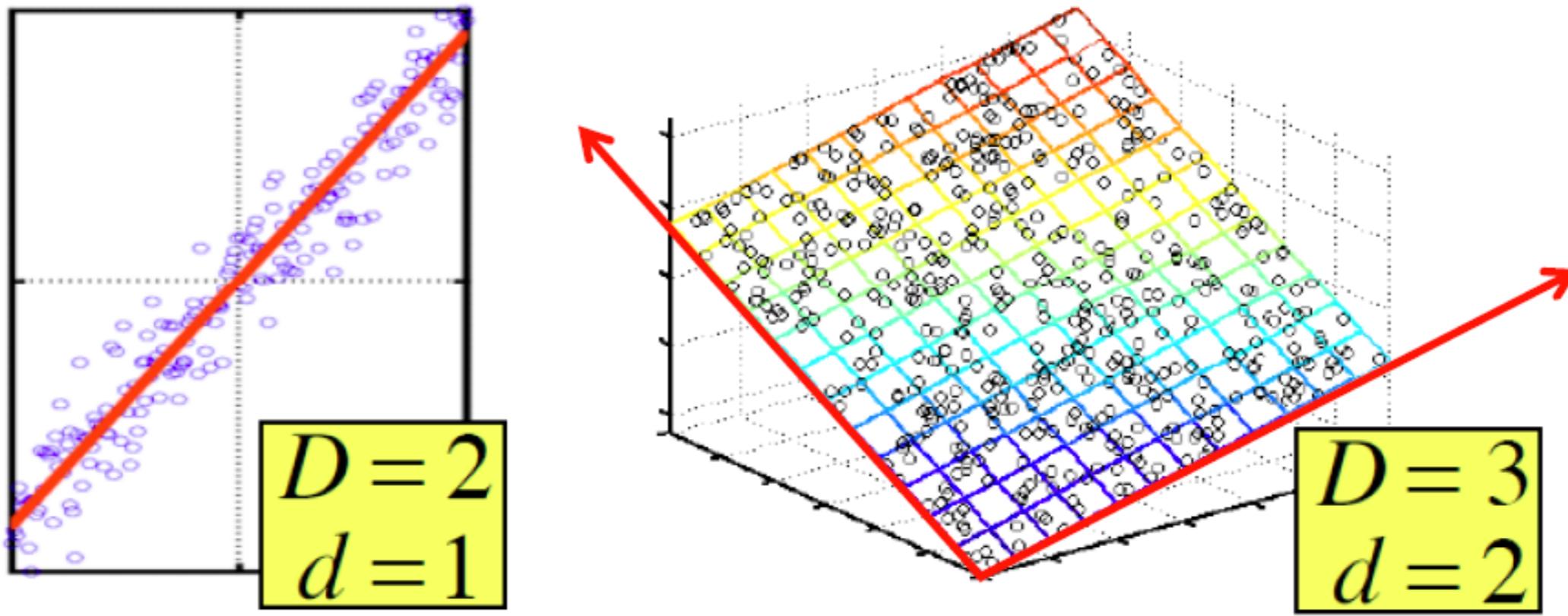
- **Assumption:** Our  $D$ -dimensional word vectors are really in a  $d$ -dimensional subspace, where  $d < D$

# Dimensionality Reduction



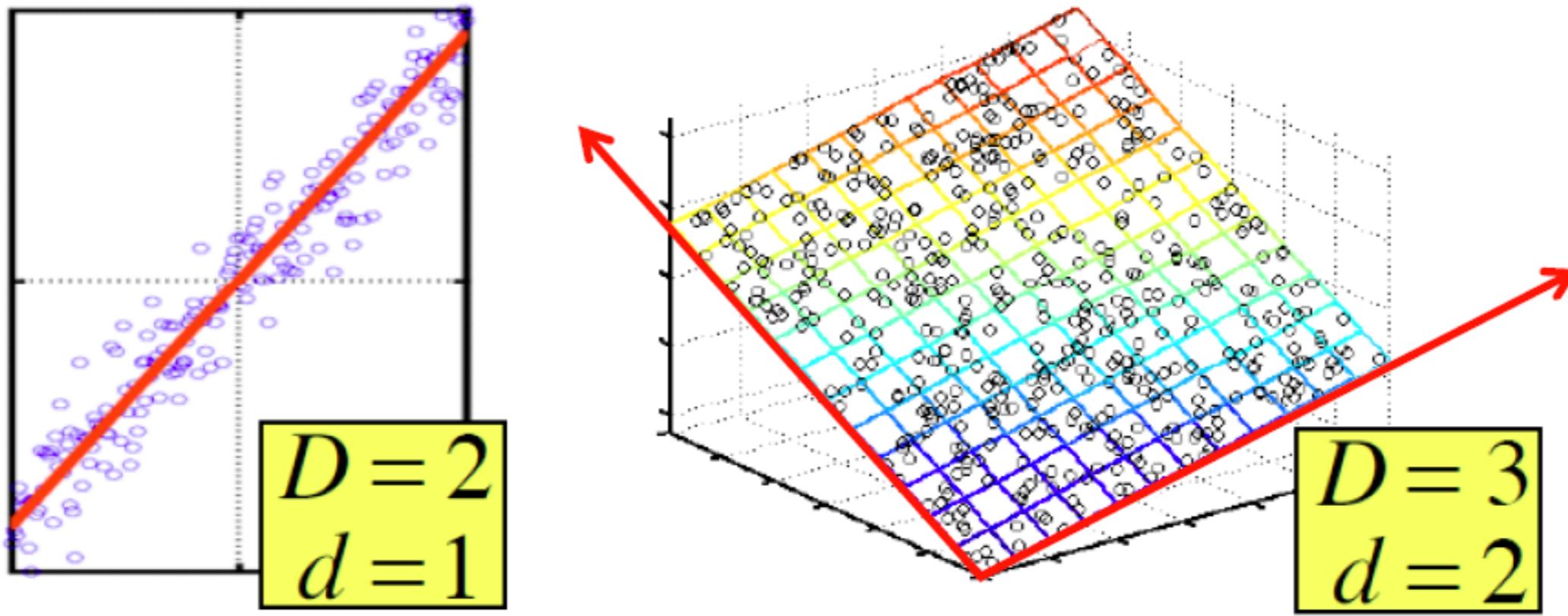
- **Assumption:** Our  $D$ -dimensional word vectors are really in a  $d$ -dimensional subspace, where  $d < D$ 
  - Why might this be true?

# Dimensionality Reduction



- **Assumption:** Our  $D$ -dimensional word vectors are really in a  $d$ -dimensional subspace, where  $d < D$ 
  - Why might this be true?
    - Consider how many synonyms there are. Do we need to count co-occurrences for each?

# Dimensionality Reduction



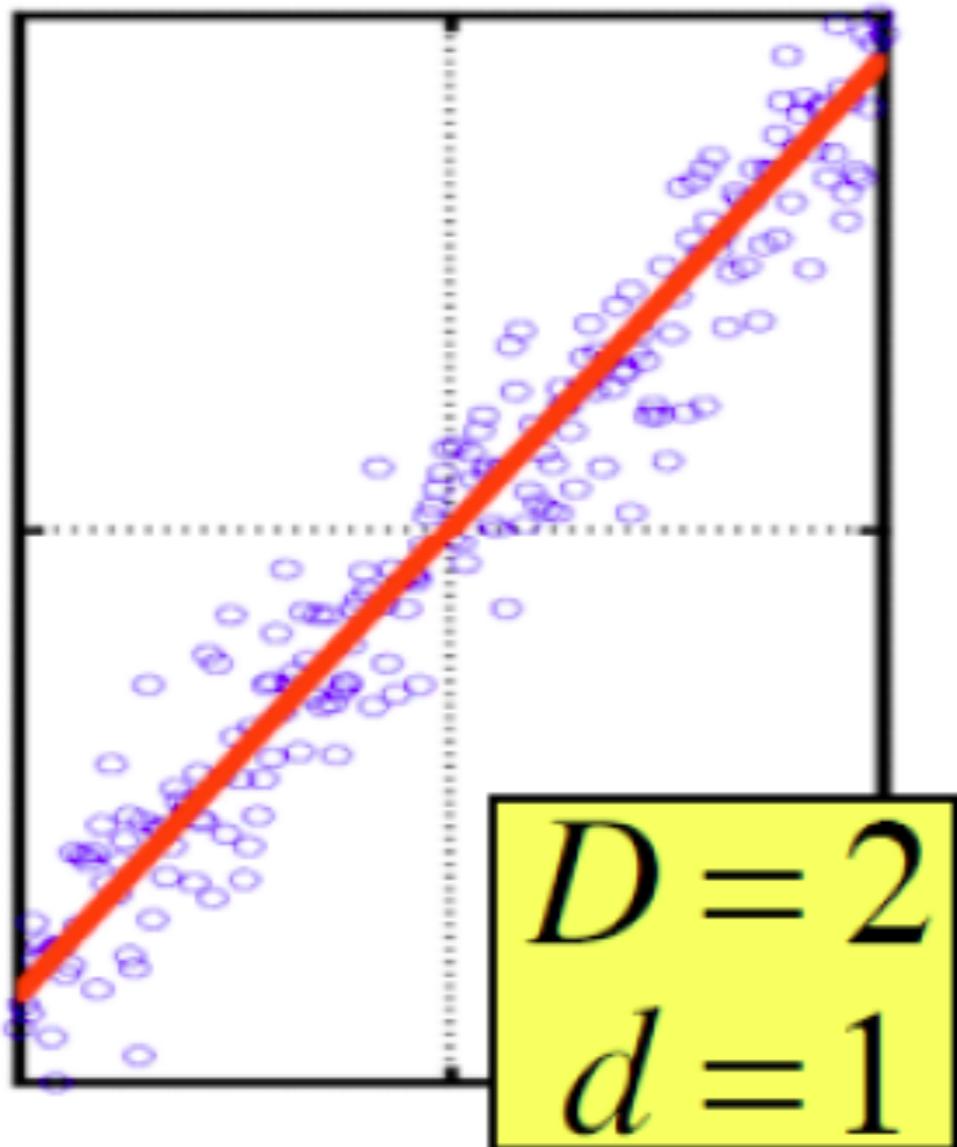
- **Assumption:** Our  $D$ -dimensional word vectors are really in a  $d$ -dimensional subspace, where  $d < D$ 
  - Why might this be true?
    - Consider how many synonyms there are. Do we need to count co-occurrences for each?
  - A lower dimensional representation of the vectors captures most of the knowledge!

# How many dimensions do we need?

	good	bad	great	awesome	hate
action	3	9	1	1	14
vampires	0	14	0	1	19
noir	6	0	3	1	0
oscar	5	0	2	4	0
drama	9	1	2	1	3
dumbledore	14	0	9	8	2
romcom	1	15	1	1	11

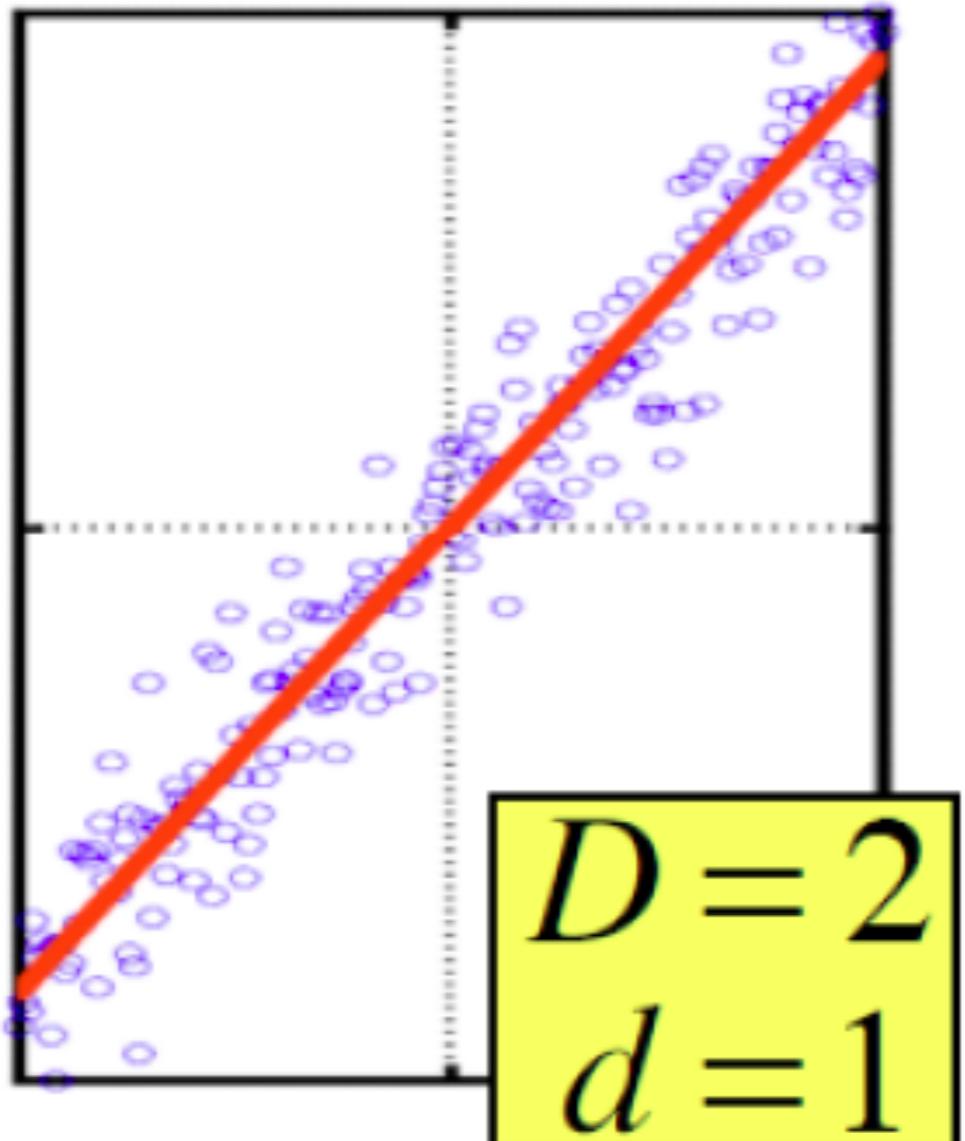
# Dimensionality Reduction

- Goal of dimensionality reduction is to discover the latent axis of data!



# Dimensionality Reduction

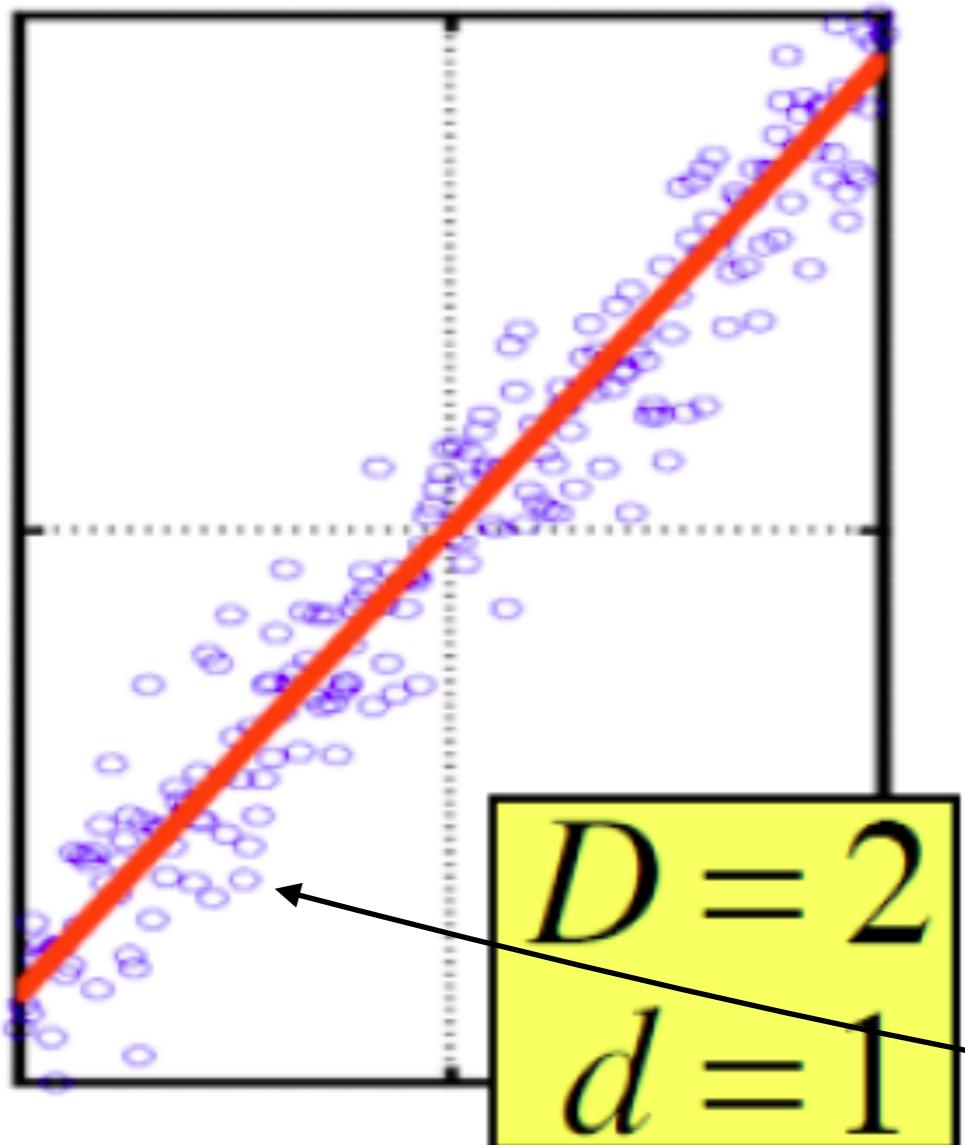
- Goal of dimensionality reduction is to discover the latent axis of data!



- Most data doesn't perfectly fit into a lower number of dimensions
- For example, the data to the left has rank 2, even though most of the data falls along a single line
- Thus, in most cases, dimensionality reduction will add error to the representations, e.g., the points do not fall exactly lie on the axis

# Dimensionality Reduction

- Goal of dimensionality reduction is to discover the latent axis of data!



- Most data doesn't perfectly fit into a lower number of dimensions
- For example, the data to the left has rank 2, even though most of the data falls along a single line
- Thus, in most cases, dimensionality reduction will add error to the representations, e.g., the points do not fall exactly lie on the axis



# **Latent Semantic Analysis**

# Latent semantic analysis

- Latent Semantic Analysis/Indexing (Deerwester et al. 1998) is this process of applying SVD to the term-document co-occurrence matrix
- Terms typically weighted by tf-idf
- This is a form of dimensionality reduction (for terms, from a D-dimensional sparse vector to a K-dimensional dense one),  $K \ll D$ .

# Singular Value Decomposition: Definition

$$A_{[m \times n]} = U_{[m \times r]} \sum_{[r \times r]} (V_{[n \times r]})^T$$

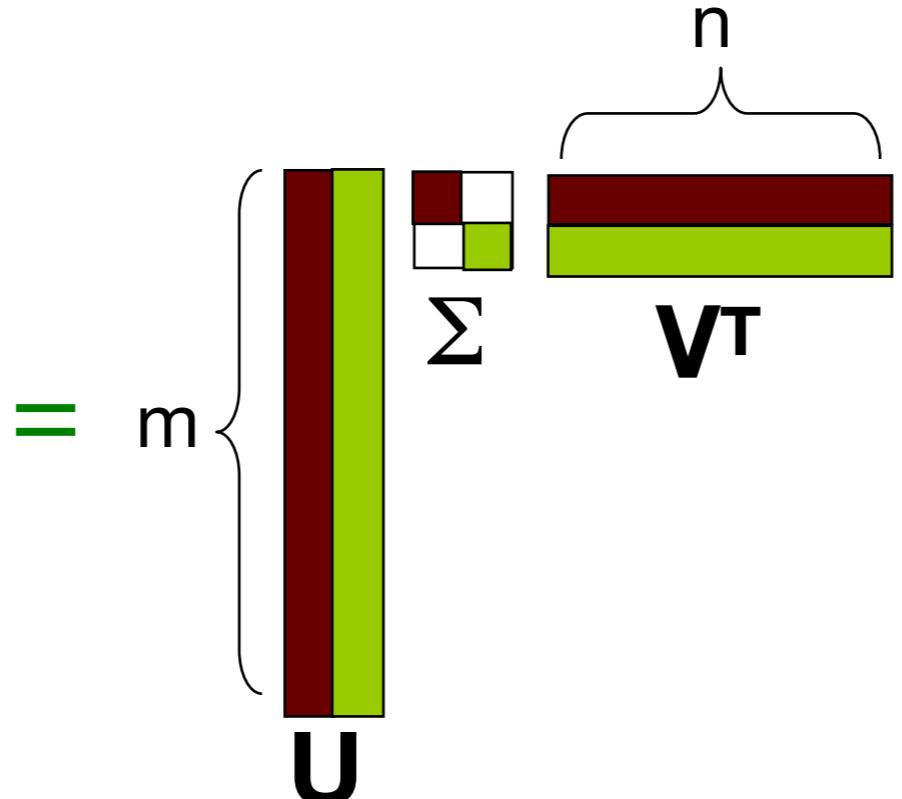
# Singular Value Decomposition: Definition

$$A_{[m \times n]} = U_{[m \times r]} \sum_{[r \times r]} (V_{[n \times r]})^T$$

- A: Input data matrix
  - $m \times n$  matrix (e.g., m documents, n terms)
- U: Left singular vectors
  - $m \times r$  matrix (m documents, r concepts)
- $\Sigma$ : Singular values
  - $r \times r$  diagonal matrix (strength of each ‘concept’)  
( $r$  : rank of the matrix A)
- V: Right singular vectors
  - $n \times r$  matrix (n terms, r concepts)

# SVD Example: Term-Document Matrix

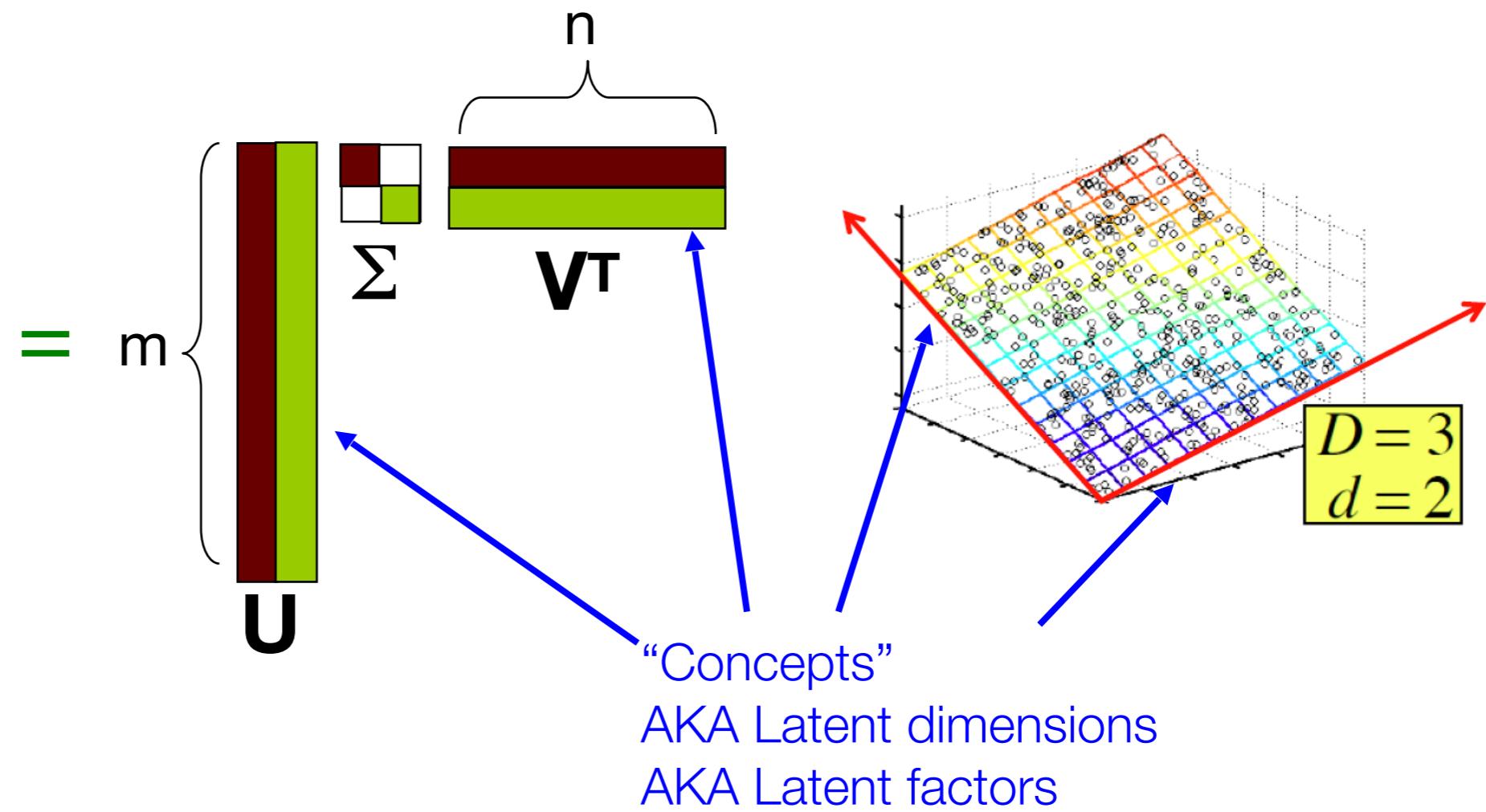
$$A = U \Sigma V^T$$

$$\begin{array}{c} \begin{matrix} & \text{Doc 1} & \text{Doc 2} & \text{Doc 3} & \text{Doc 4} & \text{Doc 5} \\ \text{Sports} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \\ \text{Terms} \\ \downarrow \end{matrix} \\ \begin{matrix} & \text{Science} \\ \text{Science} & \text{Terms} \\ \downarrow \end{matrix} \end{array} = m \left\{ \begin{matrix} U & \Sigma & V^T \\ \text{m} & \text{n} \end{matrix} \right\}$$


# SVD Example: Term-Document Matrix

$$A = U \Sigma V^T$$

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
Sports Terms	1	1	1	0	0
Science Terms	0	2	0	4	4
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
	0	0	0	5	5
	0	1	0	2	2



# SVD Example: Term-Document Matrix

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

↑  
Sports  
Terms  
↓  
↑  
Science  
Terms  
↓

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

# SVD Example: Term-Document Matrix

$$A = U \Sigma V^T$$

↑ Sports Terms ↓      ↑ Science Terms ↓

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
1	1	1	1	0	0
3	3	3	3	0	0
4	4	4	4	0	0
5	5	5	5	0	0
0	2	0	4	4	4
0	0	0	5	5	5
0	1	0	2	2	2

=

0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
0.68	0.11	-0.05
0.15	-0.59	0.65
0.07	-0.73	-0.67
0.07	-0.29	0.32

 $\times$ 

12.4	0	0
0	9.5	0
0	0	1.3

 $\times$ 

0.56	0.59	0.56	0.09	0.09
0.12	-0.02	0.12	-0.69	-0.69
0.40	-0.80	0.40	0.09	0.09

# SVD Example: Term-Document Matrix

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
↑ Sports Terms ↓	1 1 1 0 0	3 3 3 0 0	4 4 4 0 0	5 5 5 0 0	0 2 0 4 4
↑ Science Terms ↓	0 0 0 5 5	0 0 0 5 5	0 1 0 2 2		

=

0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
<b>0.68</b>	0.11	-0.05
0.15	<b>-0.59</b>	<b>0.65</b>
0.07	<b>-0.73</b>	<b>-0.67</b>
0.07	<b>-0.29</b>	<b>0.32</b>

x

<b>12.4</b>	0	0
0	<b>9.5</b>	0
0	0	<b>1.3</b>

x

Sports-concept      Science-concept

**U** is “word-to-concept” similarity matrix

0.56	0.59	0.56	0.09	0.09
0.12	-0.02	0.12	<b>-0.69</b>	<b>-0.69</b>
0.40	<b>-0.80</b>	0.40	0.09	0.09

# SVD Example: Term-Document Matrix

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
↑ Sports Terms ↓	1 1 1 0 0	3 3 3 0 0	4 4 4 0 0	5 5 5 0 0	0 2 0 4 4
↑ Science Terms ↓	0 0 0 5 5	0 0 0 5 5	0 1 0 2 2		

$$= \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix}$$

Sports-concept      Science-concept

$\mathbf{U}$  is “word-to-concept” similarity matrix

“strength” of the sports-concept in the whole corpus

$$x \quad \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \quad x$$

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD Example: Term-Document Matrix

$$A = U \Sigma V^T$$

	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
↑ Sports Terms ↓	1 1 1 0 0	3 3 3 0 0	4 4 4 0 0	5 5 5 0 0	0 2 0 4 4
↑ Science Terms ↓	0 0 0 5 5	0 0 0 2 2			

=

0.13	0.02	-0.01
0.41	0.07	-0.03
0.55	0.09	-0.04
<b>0.68</b>	0.11	-0.05
0.15	<b>-0.59</b>	<b>0.65</b>
0.07	<b>-0.73</b>	<b>-0.67</b>
0.07	<b>-0.29</b>	<b>0.32</b>

**U** is “word-to-concept” similarity matrix

“strength” of the sports-concept in the whole corpus

$$x \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} x$$

**V** is the “document-to-concept” matrix

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# What does the SVD give us?

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

- $\mathbf{U} \Sigma$ : Gives the coordinates of the points in the projection axis

1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

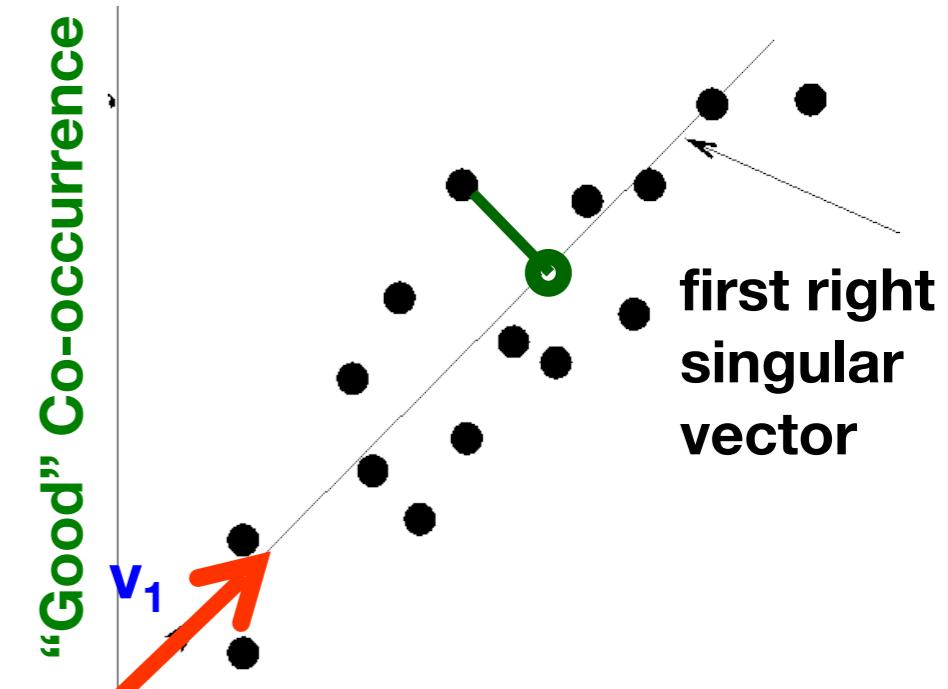
**Projection of words  
on the “Sports” axis**  
 $(\mathbf{U} \Sigma)^T$ :

1.61	0.19	-0.01
5.08	0.66	-0.03
6.82	0.85	-0.05
8.43	1.04	-0.06
1.86	-5.60	0.84
0.86	-6.93	-0.87
0.86	-2.75	0.41

# What does the SVD give us?

$$\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$$

- $\mathbf{U} \Sigma$ : Gives the coordinates of the points in the projection axis



Projection of words  
on the “Sports” axis

$(\mathbf{U} \Sigma)^T$ :

1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

“Good” Co-occurrence

“Bad” Co-occurrence

1.61	0.19	-0.01
5.08	0.66	-0.03
6.82	0.85	-0.05
8.43	1.04	-0.06
1.86	-5.60	0.84
0.86	-6.93	-0.87
0.86	-2.75	0.41

# How do we use the SVD to reduce the dimensionality?

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

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

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

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

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

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

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

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

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

The diagram illustrates the SVD decomposition of a matrix. The original matrix on the left has a row of zeros highlighted with a red double underline. The middle matrix shows the first three non-zero singular values. The right matrix shows the first two non-zero singular values. Red lines and an 'X' mark indicate that the third singular value and its corresponding column in the right matrix are being set to zero, demonstrating how dimensionality reduction is achieved by truncating the SVD.

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

# How do we use the SVD to reduce the dimensionality?

**Answer:** Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.92} & \mathbf{0.95} & \mathbf{0.92} & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ \mathbf{3.90} & \mathbf{4.04} & \mathbf{3.90} & 0.01 & 0.01 \\ \mathbf{4.82} & \mathbf{5.00} & \mathbf{4.82} & 0.03 & 0.03 \\ 0.70 & \mathbf{0.53} & 0.70 & \mathbf{4.11} & \mathbf{4.11} \\ -0.69 & 1.34 & -0.69 & \mathbf{4.78} & \mathbf{4.78} \\ 0.32 & \mathbf{0.23} & 0.32 & \mathbf{2.01} & \mathbf{2.01} \end{bmatrix}$$

The SVD is minimizing the Frobenius norm:

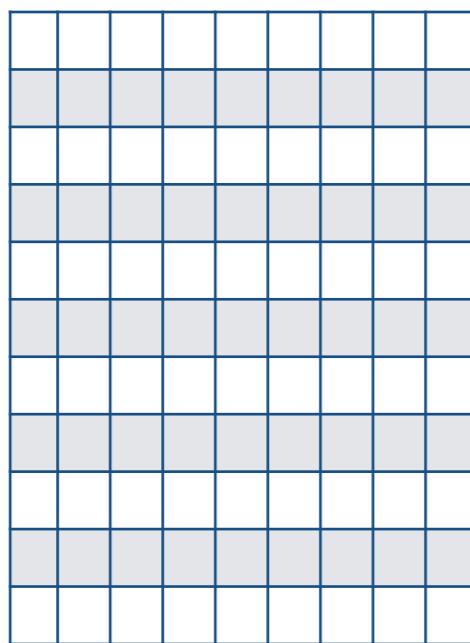
$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is “small”

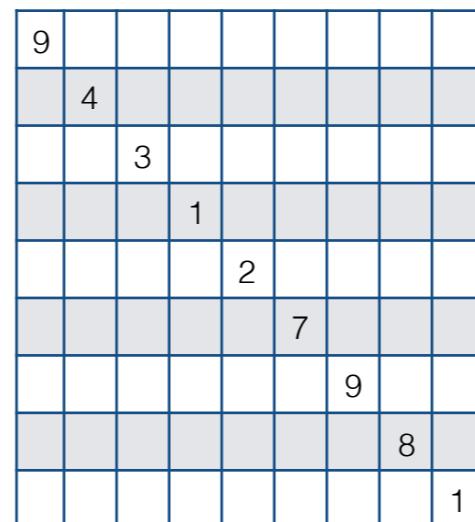
# Singular value decomposition provides a powerful tool for dimensionality reduction

- Any  $n \times p$  matrix  $X$  can be decomposed into the product of three matrices (where  $m =$  the number of linearly independent rows)



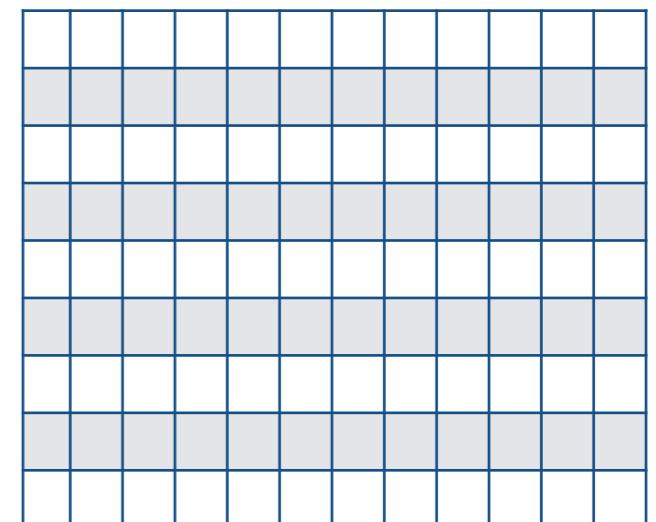
$n \times m$

$\times$



$m \times m$   
(diagonal)

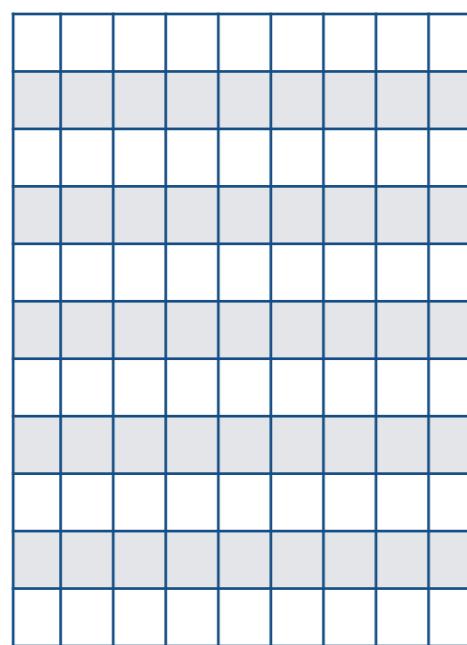
$\times$



$m \times p$

# Singular value decomposition

- We can approximate the full matrix by only considering the leftmost k terms in the diagonal matrix



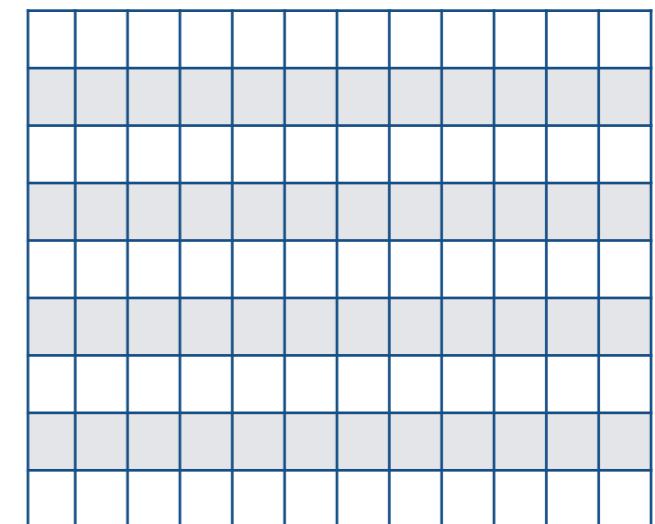
$n \times m$

$\times$

9	4					
	0					
		0				
			0			
				0		
					0	
						0

$m \times m$   
(diagonal)

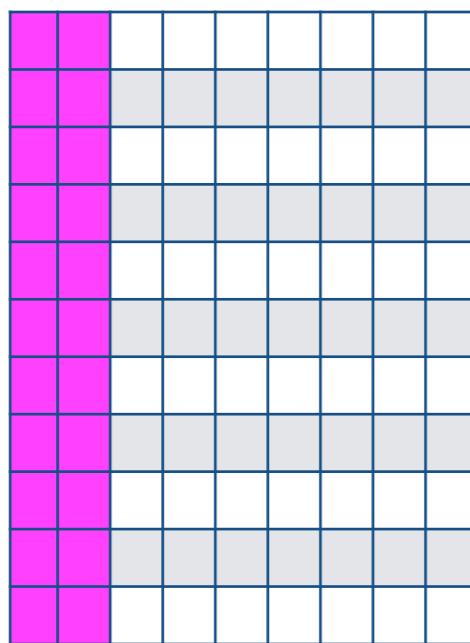
$\times$



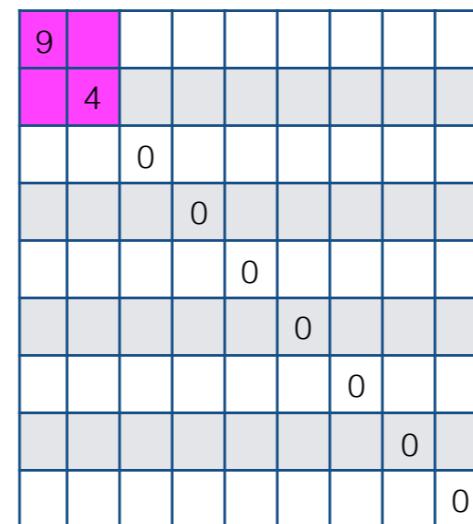
$m \times p$

# Singular value decomposition

- We can **approximate** the full matrix by only considering the leftmost  $k$  terms in the diagonal matrix (the  $k$  largest singular values)

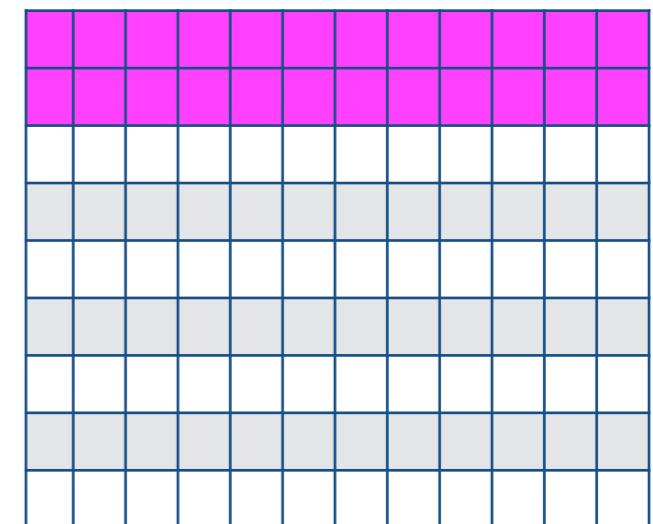



$\times$



9	4	0	0	0
		0	0	0
			0	0
				0

$\times$




$n \times m$

$m \times m$

$m \times p$

	Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear
knife	1	1	4	2		2		2
dog	2		6	6		2		12
sword	17	2	7	12		2		17
love	64		135	63		12		48
like	75	38	34	36	34	41	27	44

U

knife		
dog		
sword		
love		
like		

$\Sigma$


V

Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear

Low-dimensional  
representation for  
terms (here 2-dim)



knife		
dog		
sword		
love		
like		

$\Sigma$


Low-dimensional  
representation for  
documents (here 2-dim)



Hamlet	Macbeth	Romeo & Juliet	Richard III	Julius Caesar	Tempest	Othello	King Lear

# Returning to this example

	good	bad	great	awesome	hate
action	3	9	1	1	14
vampires	0	14	0	1	19
noir	6	0	3	1	0
oscar	5	0	2	4	0
drama	9	1	2	1	3
dumbledo	14	0	9	8	2
romcom	1	15	1	1	11

# Returning to this example

SVD

	good	bad	great	awesome	hate
action	3	9	1	1	14
vampires	0	14	0	1	19
noir	6	0	3	1	0
oscar	5	0	2	4	0
drama	9	1	2	1	3
dumbledo	14	0	9	8	2
romcom	1	15	1	1	11

=

U

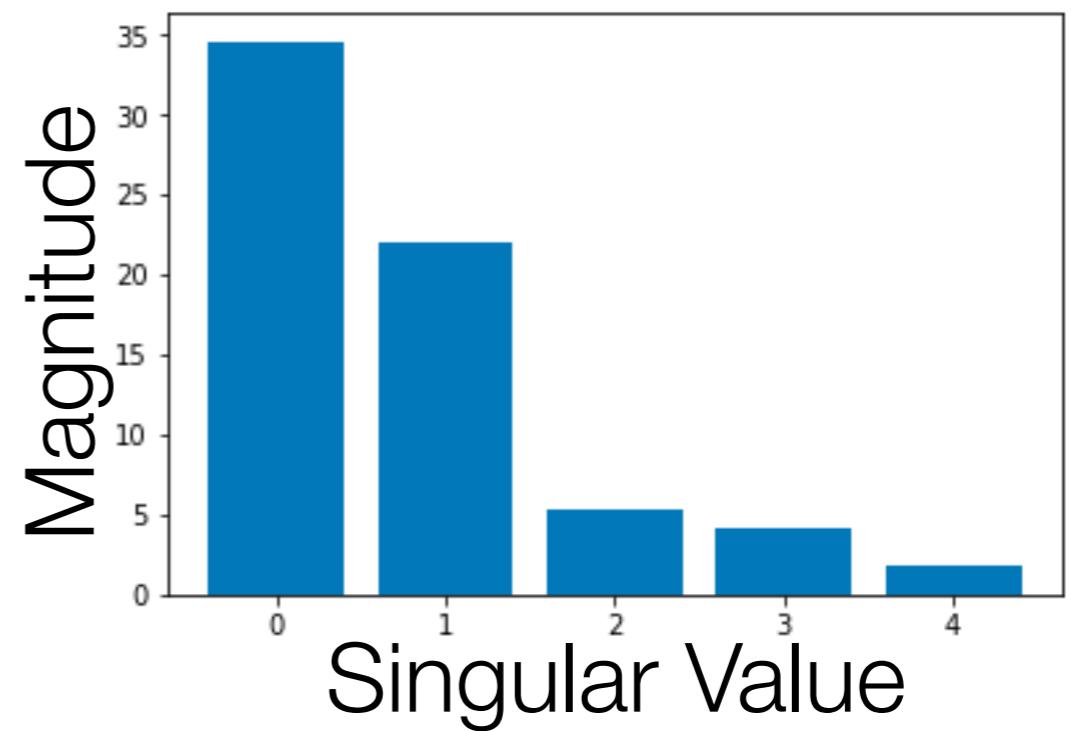
	good	bad
action	-0.49	0
vampires	-0.67	0.2
noir	-0.04	-0.29
oscar	-0.04	-0.29
drama	-0.13	-0.36
dumbledo	-0.15	-0.81
romcom	-0.52	0.11

=

U

	good	bad
action	-0.49	0
vampires	-0.67	0.2
noir	-0.04	-0.29
oscar	-0.04	-0.29
drama	-0.13	-0.36
dumbledo	-0.15	-0.81
romcom	-0.52	0.11

$\Sigma$



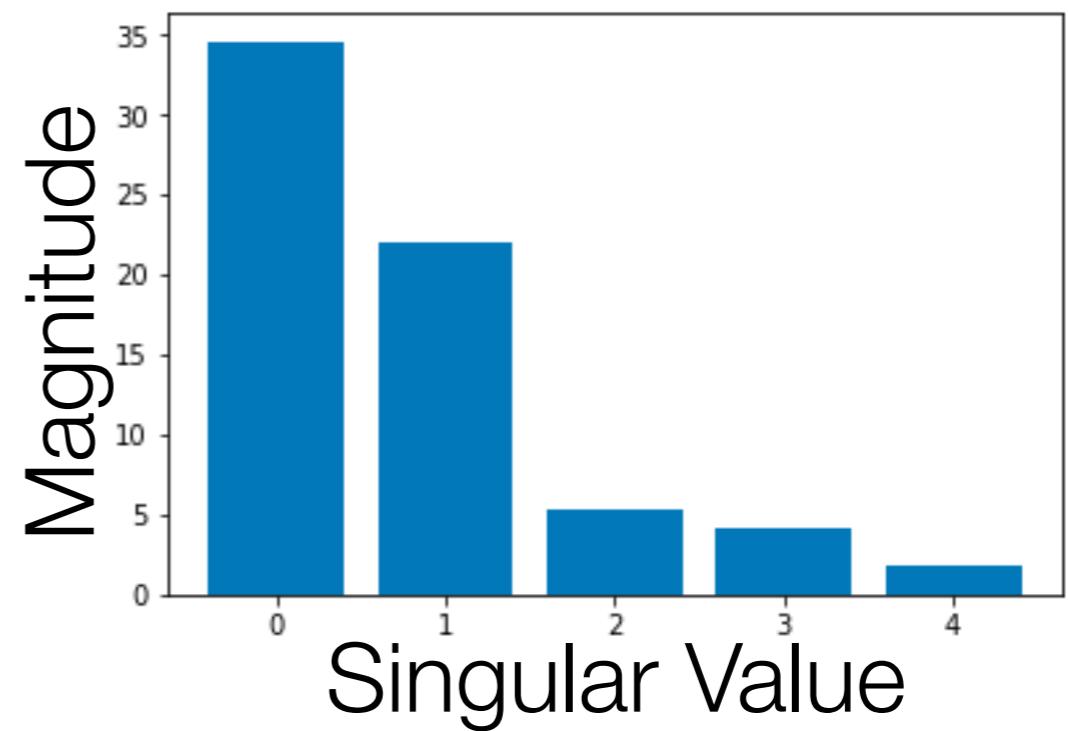
=

U

	good	bad
action	-0.49	0
vampires	-0.67	0.2
noir	-0.04	-0.29
oscar	-0.04	-0.29
drama	-0.13	-0.36
dumbledo	-0.15	-0.81
romcom	-0.52	0.11

$\Sigma$

(V not  
shown)



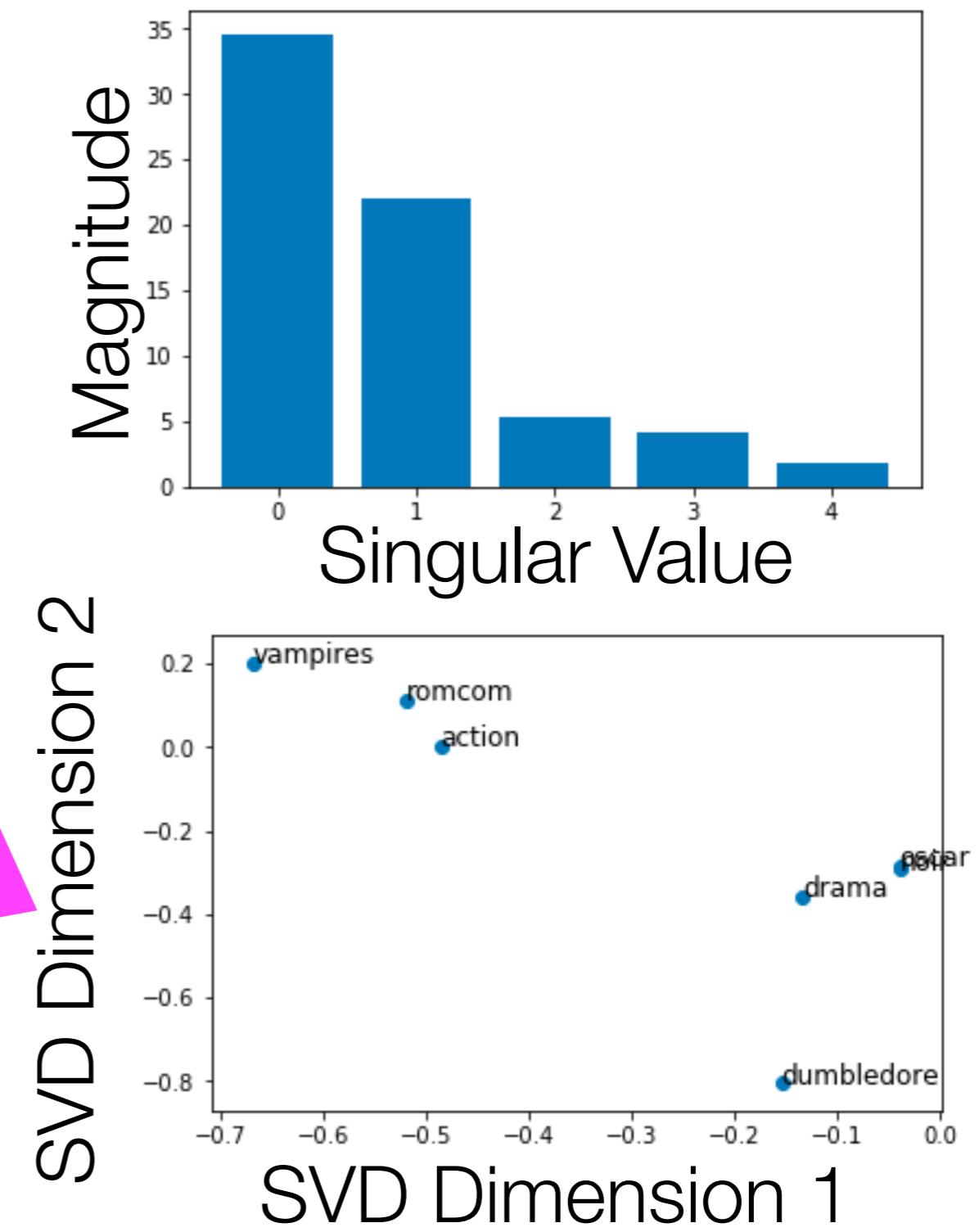
=

U

	good	bad
action	-0.49	0
vampires	-0.67	0.2
noir	-0.04	-0.29
oscar	-0.04	-0.29
drama	-0.13	-0.36
dumbledore	-0.15	-0.81
romcom	-0.52	0.11

$\Sigma$

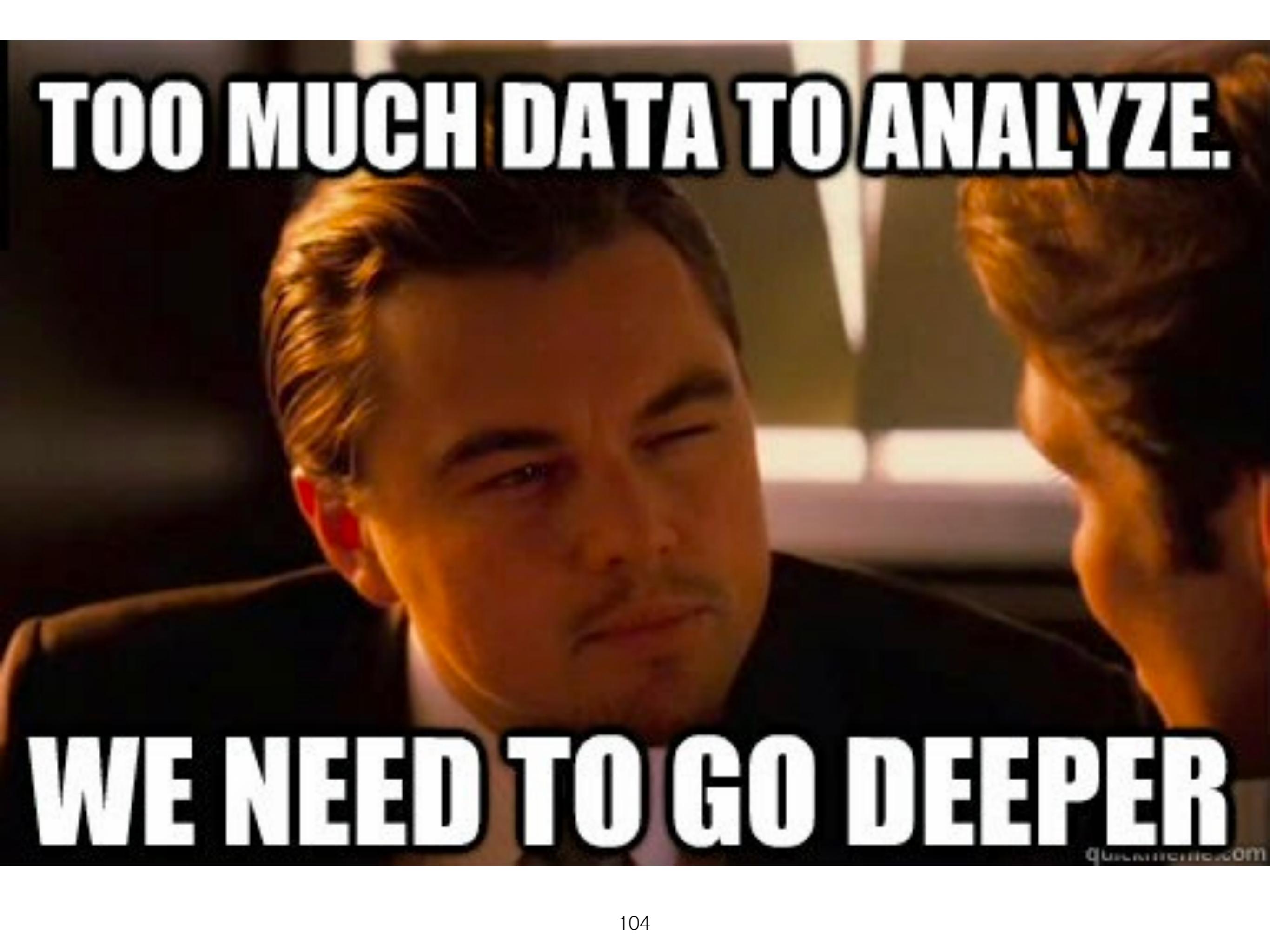
(V not shown)



# Problems with SVD

$$\begin{array}{c} m \\ n \end{array} \boxed{X} = \begin{array}{c} r \\ n \end{array} \boxed{U} \begin{array}{c} r \\ r \end{array} \boxed{S} \begin{array}{c} m \\ r \end{array} \boxed{V^T}$$
$$\begin{array}{c} m \\ n \end{array} \boxed{\hat{X}} = \begin{array}{c} k \\ n \end{array} \boxed{\hat{U}} \begin{array}{c} k \\ k \end{array} \boxed{\hat{S}} \begin{array}{c} m \\ k \end{array} \boxed{\hat{V}^T}$$

- Computational cost scales quadratically for  $n \times m$  matrix:  $O(mn^2)$  flops (when  $n < m$ )
- Hard to incorporate new words or documents

A close-up photograph of a man with short, light-colored hair. He is wearing a dark, collared shirt. His eyes are closed, and he has a contemplative expression. The background is blurred, showing what appears to be an office or library setting with bookshelves.

**TOO MUCH DATA TO ANALYZE.**

**WE NEED TO GO DEEPER**

© www.mrholoway.com

# **word2vec: deep learning for word representations**

# Dense vectors from prediction

- Learning low-dimensional representations of words by framing a predicting task: using context to predict words in a surrounding window
- Transform this into a *supervised* prediction problem; similar to language modeling but we're ignoring order within the context window

# Word2vec

Popular embedding method

Very fast to train

Code available on the web

Idea: **predict** rather than **count**

# 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"?

We don't actually care about this task

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

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

A word *s* near *apricot* acts as gold ‘correct answer’ to the question

“Is word *w* likely to show up near *apricot*? ”

No need for hand-labeled supervision

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

A word *s* near *apricot* acts as gold ‘correct answer’ to the question

“Is word *w* likely to show up near *apricot*? ”

No need for hand-labeled supervision

The idea comes from [neural language modeling](#)

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

A word *s* near *apricot* acts as gold ‘correct answer’ to the question

“Is word *w* likely to show up near *apricot*? ”

No need for hand-labeled supervision

The idea comes from [neural language modeling](#)

- Bengio et al. (2003)
- Collobert et al. (2011)

# What is word2vec?

# What is word2vec?

- word2vec is **not** a single algorithm

# What is word2vec?

- word2vec is **not** a single algorithm
- It is a **software package** for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - Skip-Gram
  - Various training methods
    - Negative Sampling
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words

# What is word2vec?

- word2vec is **not** a single algorithm
- It is a **software package** for representing words as vectors, containing:
  - Two distinct models
    - CBoW
    - Skip-Gram
  - Various training methods
    - Negative Sampling
    - Hierarchical Softmax
  - A rich preprocessing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting Rare Words

# Skip-gram algorithm

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 feed forward neural network to train a classifier to distinguish those two cases
4. Use the weights as the embeddings

# Dense vectors from prediction

a cocktail with **gin**  
and seltzer

X	Y
gin	a
gin	cocktail
gin	with
gin	and
gin	seltzer

Window size = 3

# Skip-Gram Training Data

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1      c2 target      c3      c4

Assume **context words** are those in +/- 2 word window

# Skip-Gram Goal

Given a tuple ( $t, c$ ) = target, context

*(apricot, jam)*

*(apricot, aardvark)*

Return probability that  $c$  is a real context word:

$$P(+|t,c)$$

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

# How to compute $p(+|t,c)$ ?

# How to compute $p(+|t,c)$ ?

Intuition:

Words are likely to appear near similar words

Model similarity with **dot-product!**

Similarity( $t,c$ )  $\propto$  target  $\cdot$  context

# How to compute $p(+|t,c)$ ?

Intuition:

Words are likely to appear near similar words

Model similarity with **dot-product!**

$\text{Similarity}(t,c) \propto \text{target} \cdot \text{context}$

Problem:

Dot product is not a probability!

(Neither is cosine)

# How to compute $p(+|t,c)$ ?

Intuition:

Words are likely to appear near similar words

Model similarity with **dot-product!**

$\text{Similarity}(t,c) \propto \text{target} \cdot \text{context}$

Problem:

Dot product is not a probability!

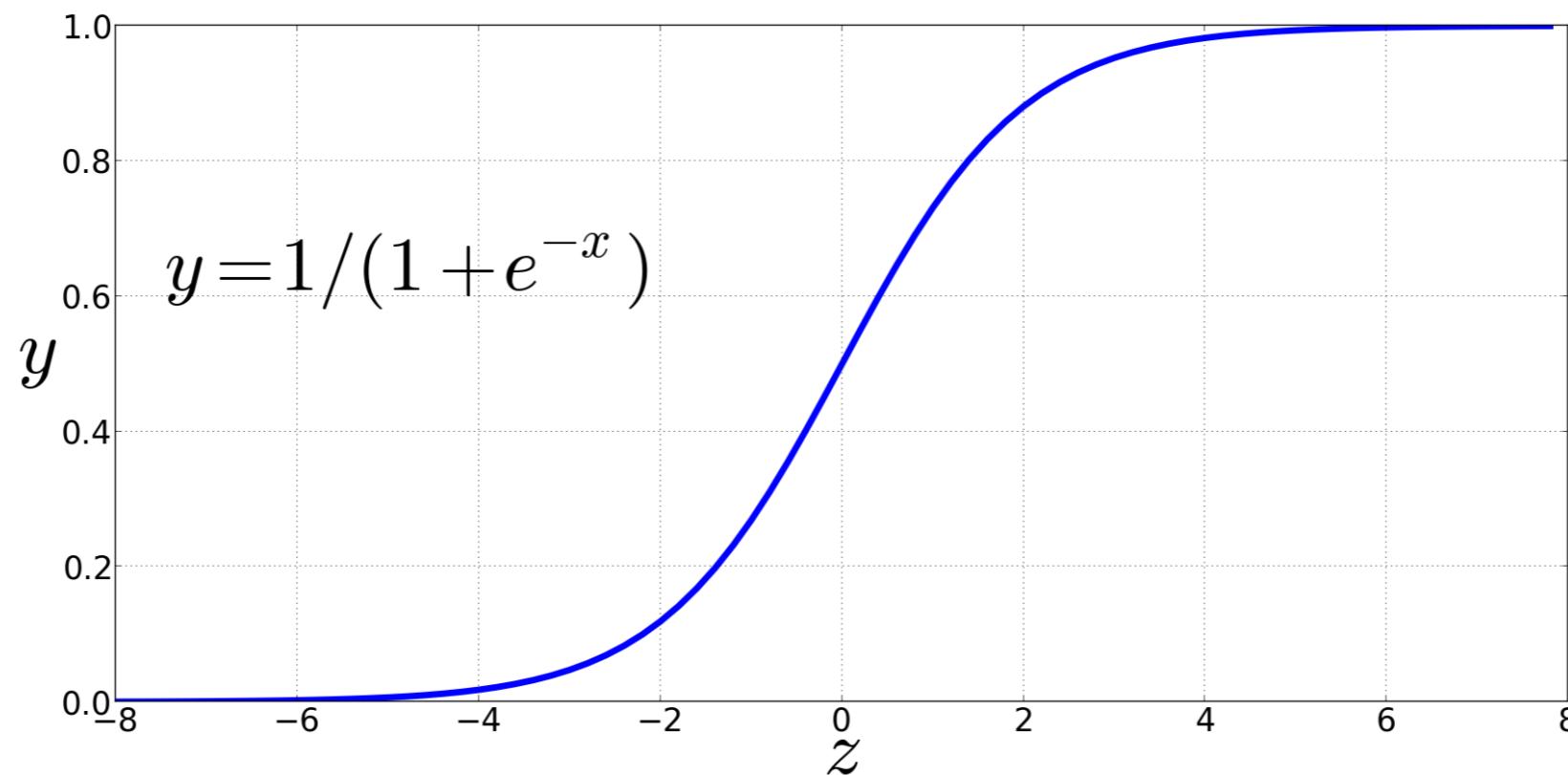
(Neither is cosine)



# Turning dot product into a probability

The sigmoid lies between 0 and 1:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



# Skip-Gram Training Data

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1            c2    t            c3    c4

Training data: input/output pairs centering on *apricot*

Asssume a +/- 2 word window

# Skip-Gram Training

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1      c2      t      c3    c4

**positive examples +**

t      c

---

apricot tablespoon

apricot of

apricot preserves

apricot or

- For each **positive example**, we'll create  $k$  **negative examples**.
- Using *noise* words
- Any random word that isn't  $t$

# Skip-Gram Training

Training sentence:

... lemon, a tablespoon of **apricot** jam a pinch ...

c1      c2      t      c3    c4

**positive examples +**

t      c

---

apricot tablespoon

apricot of

apricot preserves

apricot or

**negative examples -**

k=2

t      c      t      c

---

apricot aardvark apricot twelve

apricot puddle apricot hello

apricot where apricot dear

apricot coaxial apricot forever

# Generating positive training examples

Source Text	Training Samples
The quick brown fox jumps over the lazy dog. →	(the, quick) (the, brown)
The quick brown fox jumps over the lazy dog. →	(quick, the) (quick, brown) (quick, fox)
The quick brown fox jumps over the lazy dog. →	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown fox jumps over the lazy dog. →	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Avoid generating lots of **positive examples** for common words (“the”) by subsampling

Avoid generating lots of **positive examples** for common words (“the”) by subsampling

- Let  $z(w_i)$  be the probability of a word occurring in our corpus

Avoid generating lots of **positive examples** for common words (“the”) by subsampling

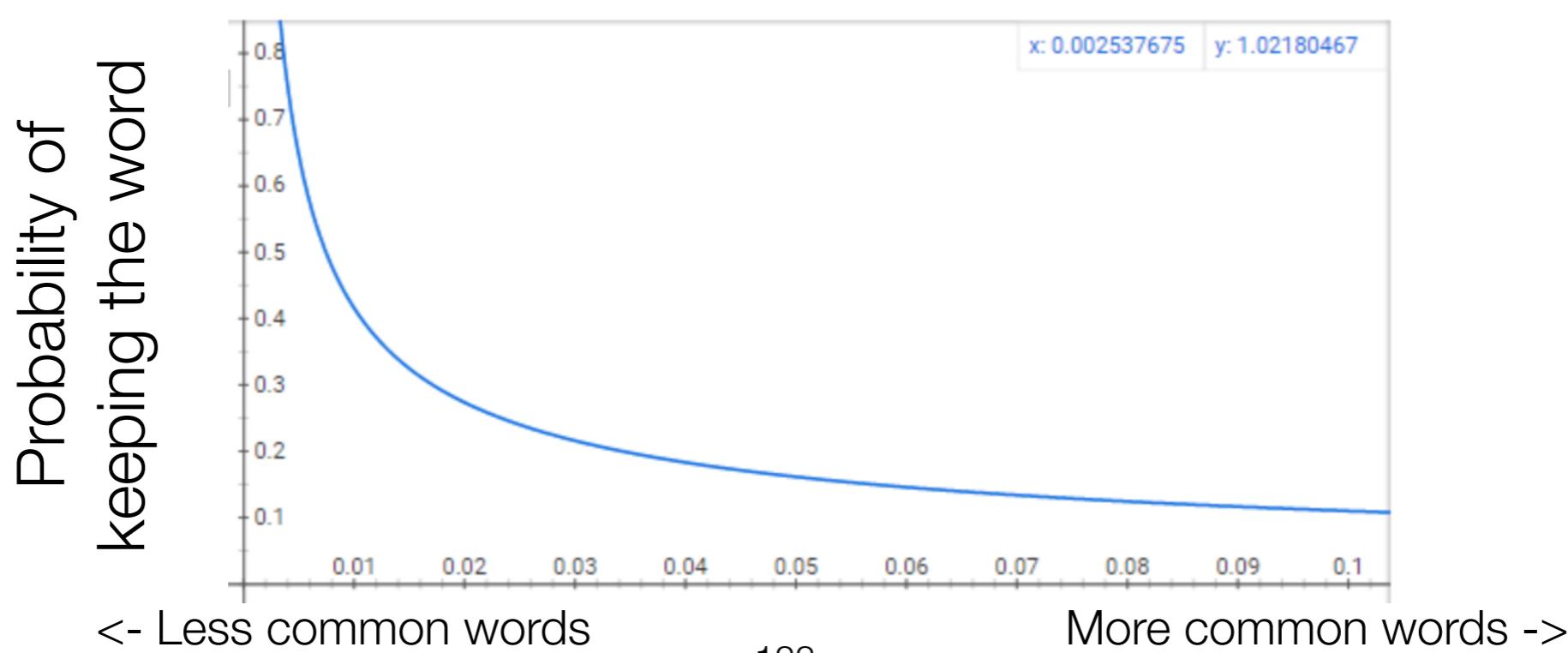
- Let  $z(w_i)$  be the probability of a word occurring in our corpus
- $P(w_i)$  is the probability of keeping a word as a training example

# Avoid generating lots of **positive examples** for common words (“the”) by subsampling

- Let  $z(w_i)$  be the probability of a word occurring in our corpus
- $P(w_i)$  is the probability of keeping a word as a training example
- The original paper defines this as 
$$P(w_i) = (\sqrt{\frac{z(w_i)}{0.001}} + 1) \cdot \frac{0.001}{z(w_i)}$$

# Avoid generating lots of **positive examples** for common words (“the”) by subsampling

- Let  $z(w_i)$  be the probability of a word occurring in our corpus
- $P(w_i)$  is the probability of keeping a word as a training example
- The original paper defines this as 
$$P(w_i) = \left( \sqrt{\frac{z(w_i)}{0.001}} + 1 \right) \cdot \frac{0.001}{z(w_i)}$$



# Prediction Task:

a cocktail with **gin** and seltzer

**x** = target  
word

**y** = Context  
word

**gin**

a

**gin**

with

**gin**

cocktail

**gin**

and

**gin**

seltzer

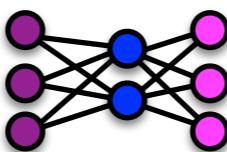
# Prediction Task:

a cocktail with **gin** and seltzer

**x** = target  
word

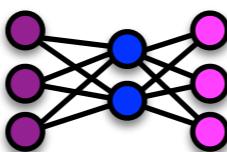
**y** = Context  
word

**gin**



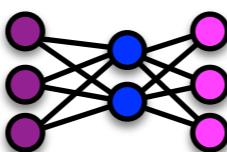
a

**gin**



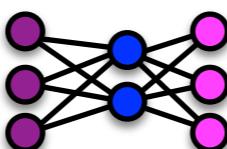
with

**gin**



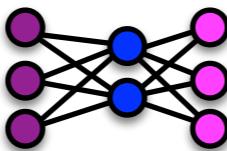
cocktail

**gin**



and

**gin**



seltzer

# Prediction Task:

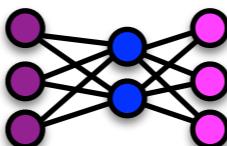
a cocktail with **gin** and seltzer

**x** = target word

**y** = Context word

**gin**

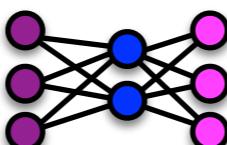
0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



**a**

**gin**

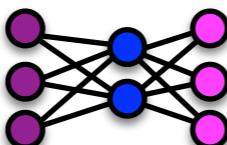
0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



**with**

**gin**

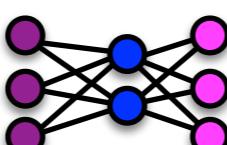
0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



**cocktail**

**gin**

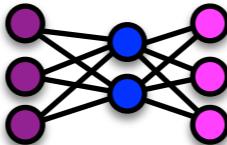
0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



**and**

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



**seltzer**

# Prediction Task:

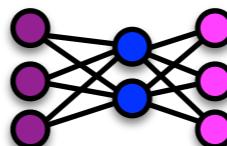
a cocktail with **gin** and seltzer

**x** = target word

**y** = Context word

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

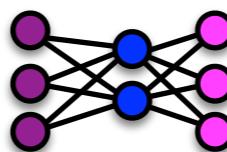


0	0	0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---

**a**

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

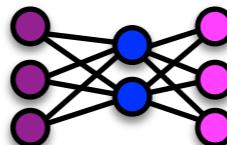


0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---

**with**

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

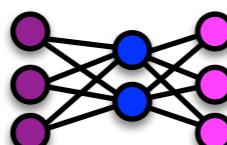


1	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**cocktail**

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

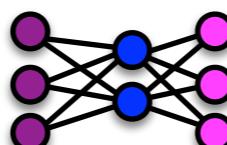


0	0	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**and**

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**seltzer**

# Prediction Task:

a cocktail with **gin** and seltzer

**x** = target word

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**gin**

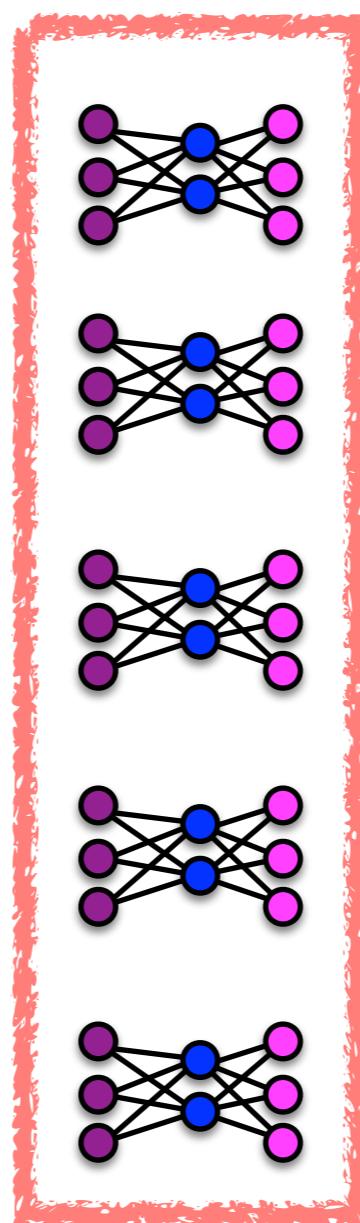
0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**gin**

0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---



0	0	0	0	0	0	0	1	0	0
---	---	---	---	---	---	---	---	---	---

**a**

0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---

**with**

1	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**cocktail**

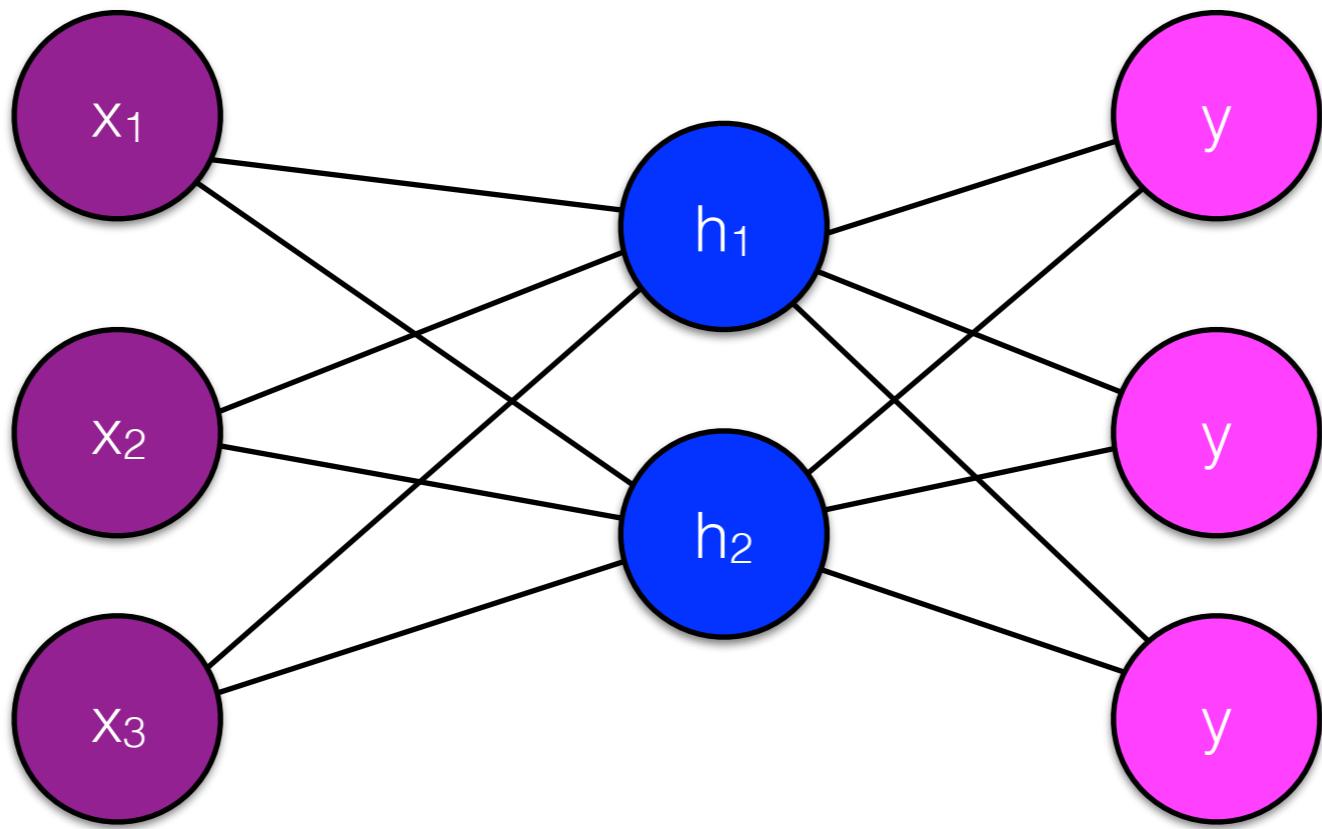
0	0	1	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**and**

0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

**seltzer**

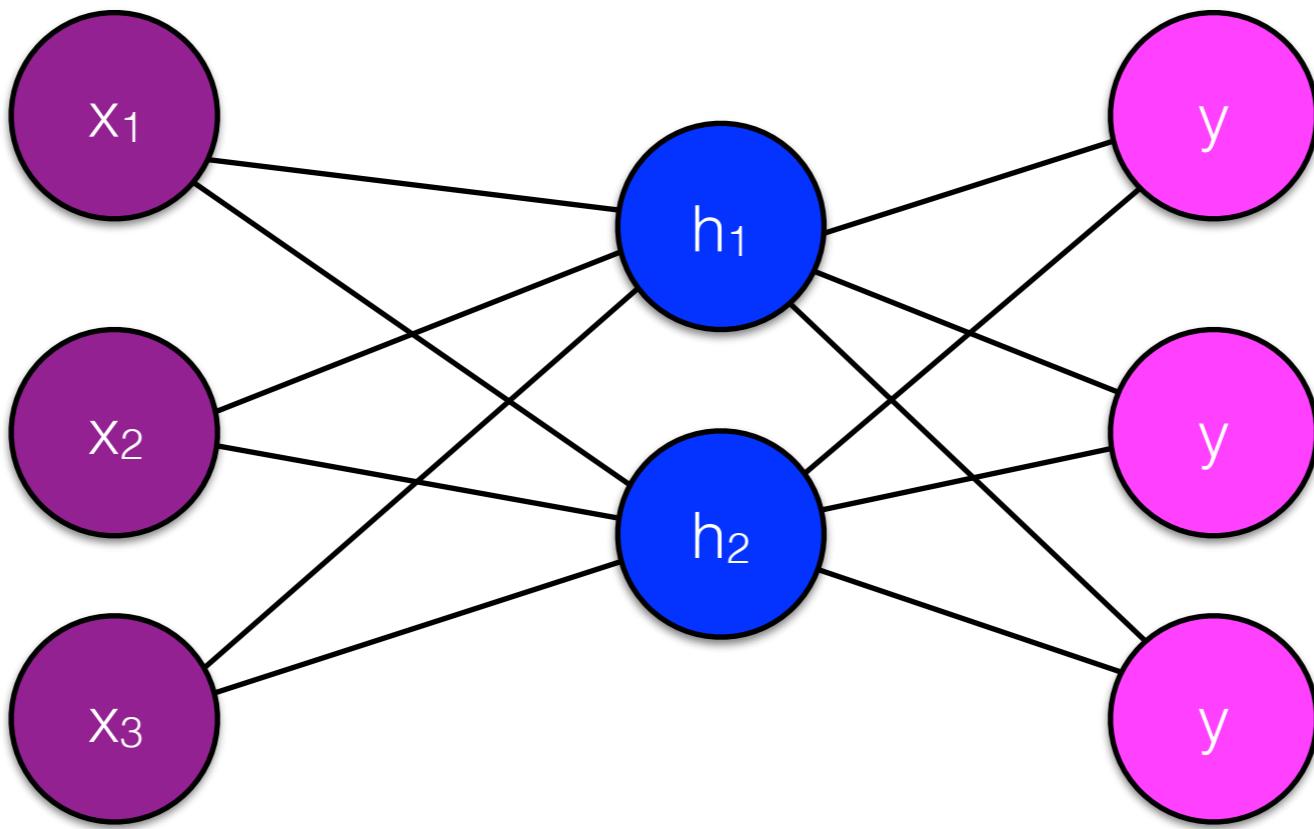
**Goal:** Learn the **weights** in a simple feed-forward neural network to do this prediction task!

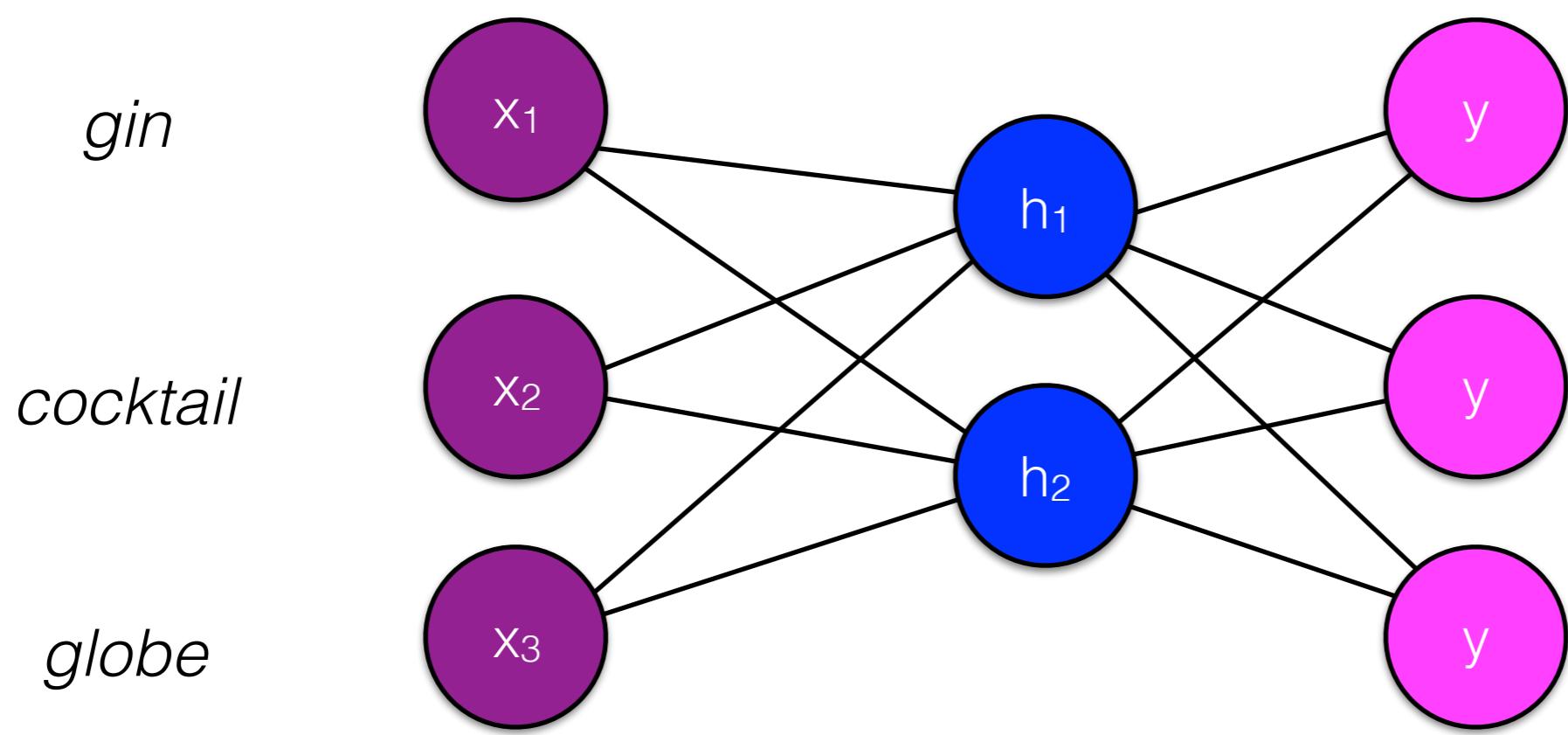


*gin*

*cocktail*

*globe*





*gin*

*cocktail*

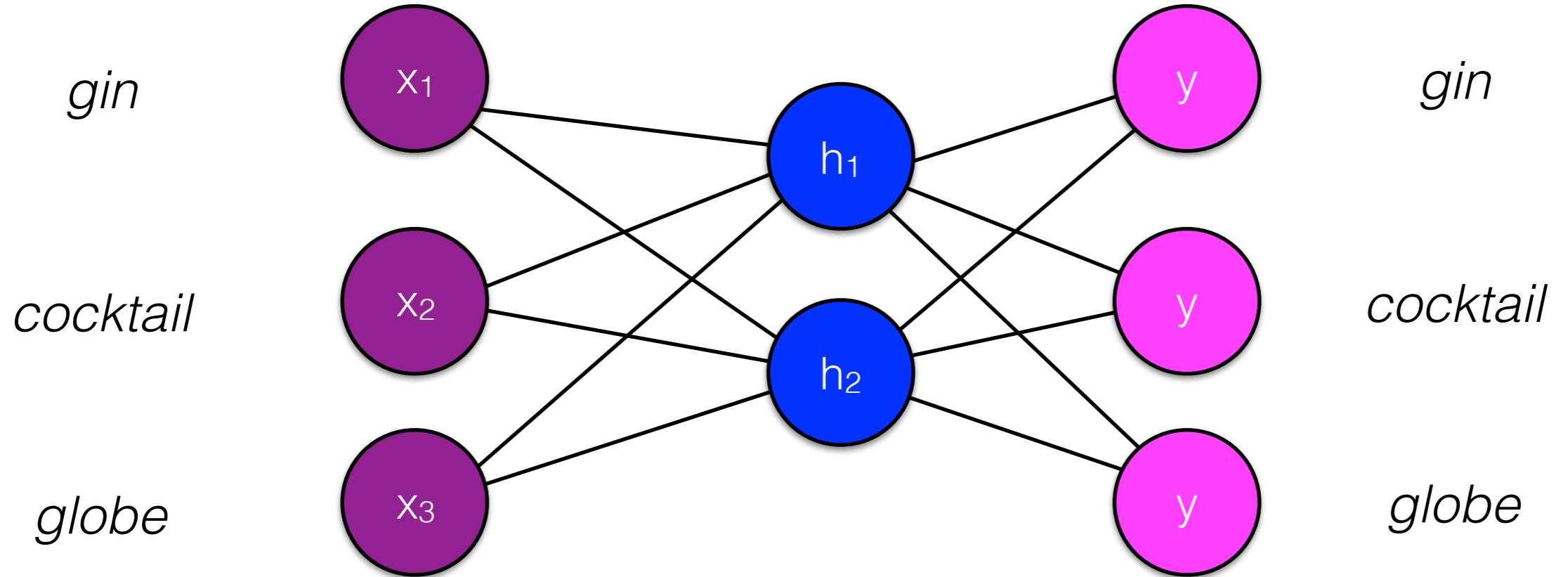
*globe*

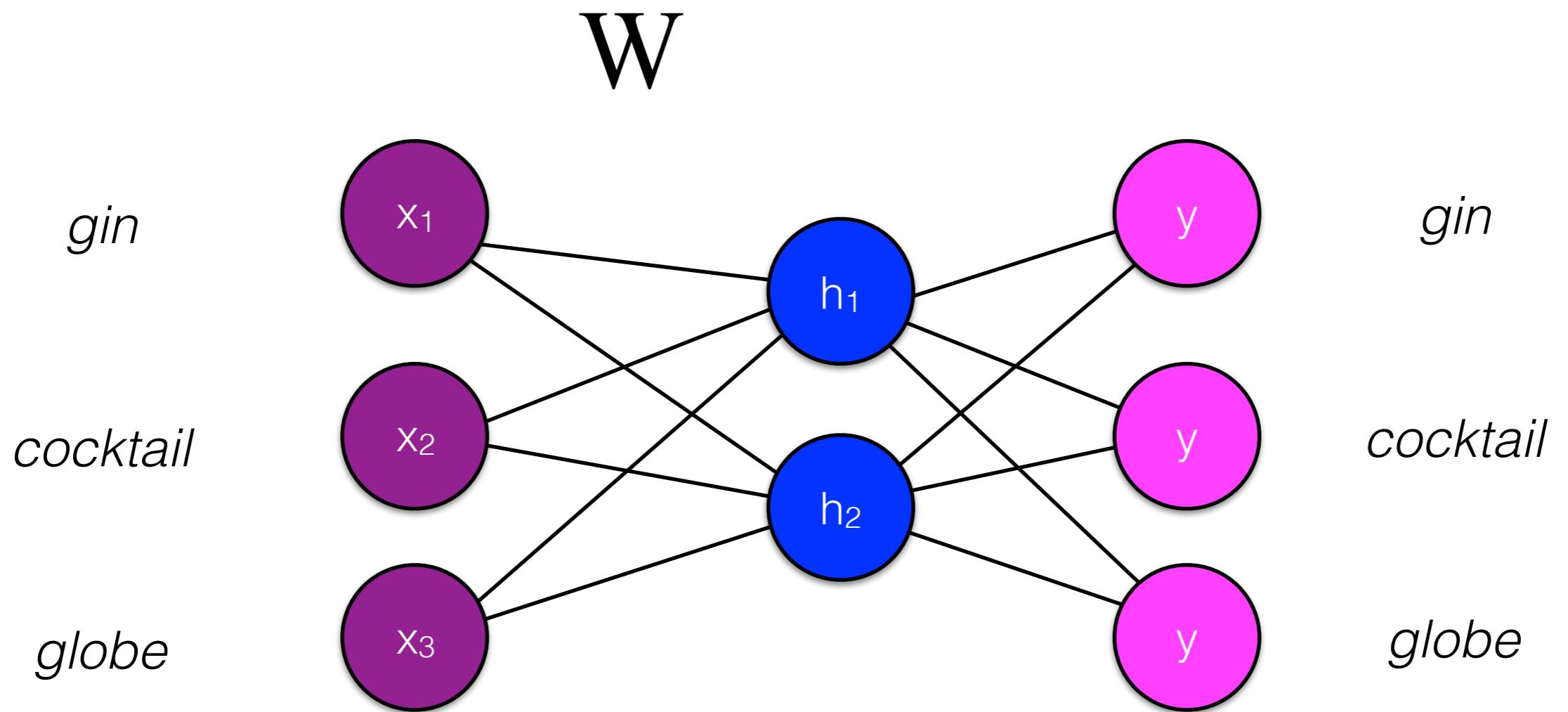
x
0
1
0

*gin*

*cocktail*

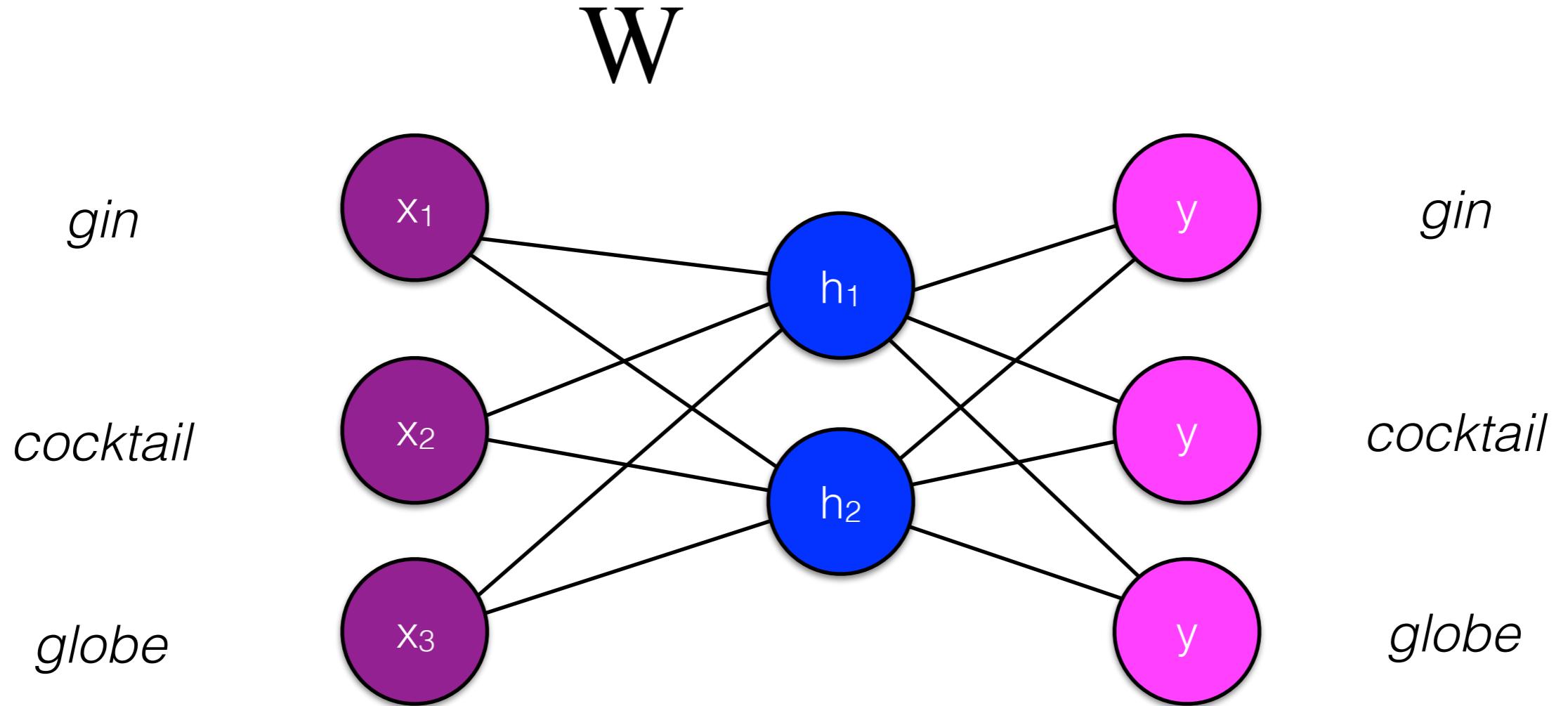
*globe*



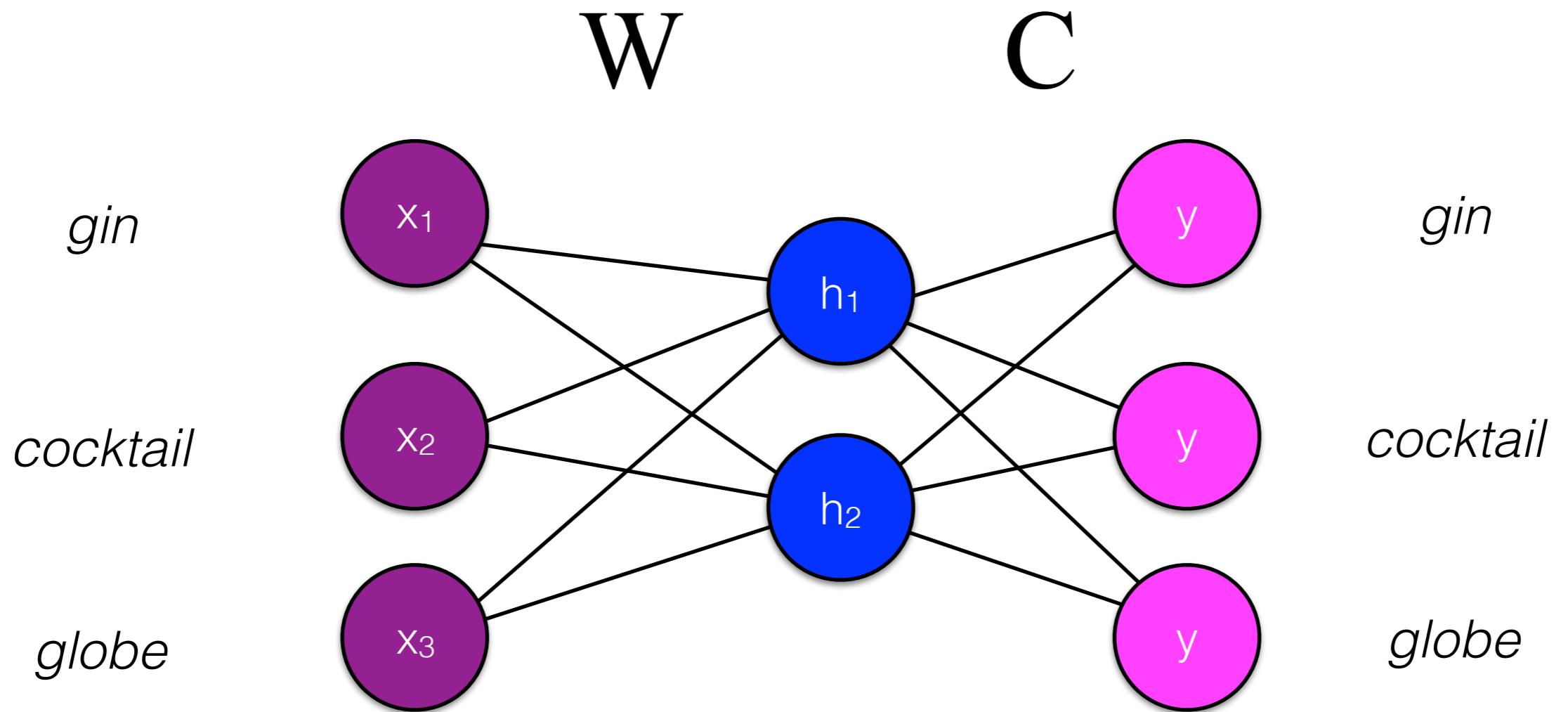


	X
<i>gin</i>	0
<i>cocktail</i>	1
<i>globe</i>	0

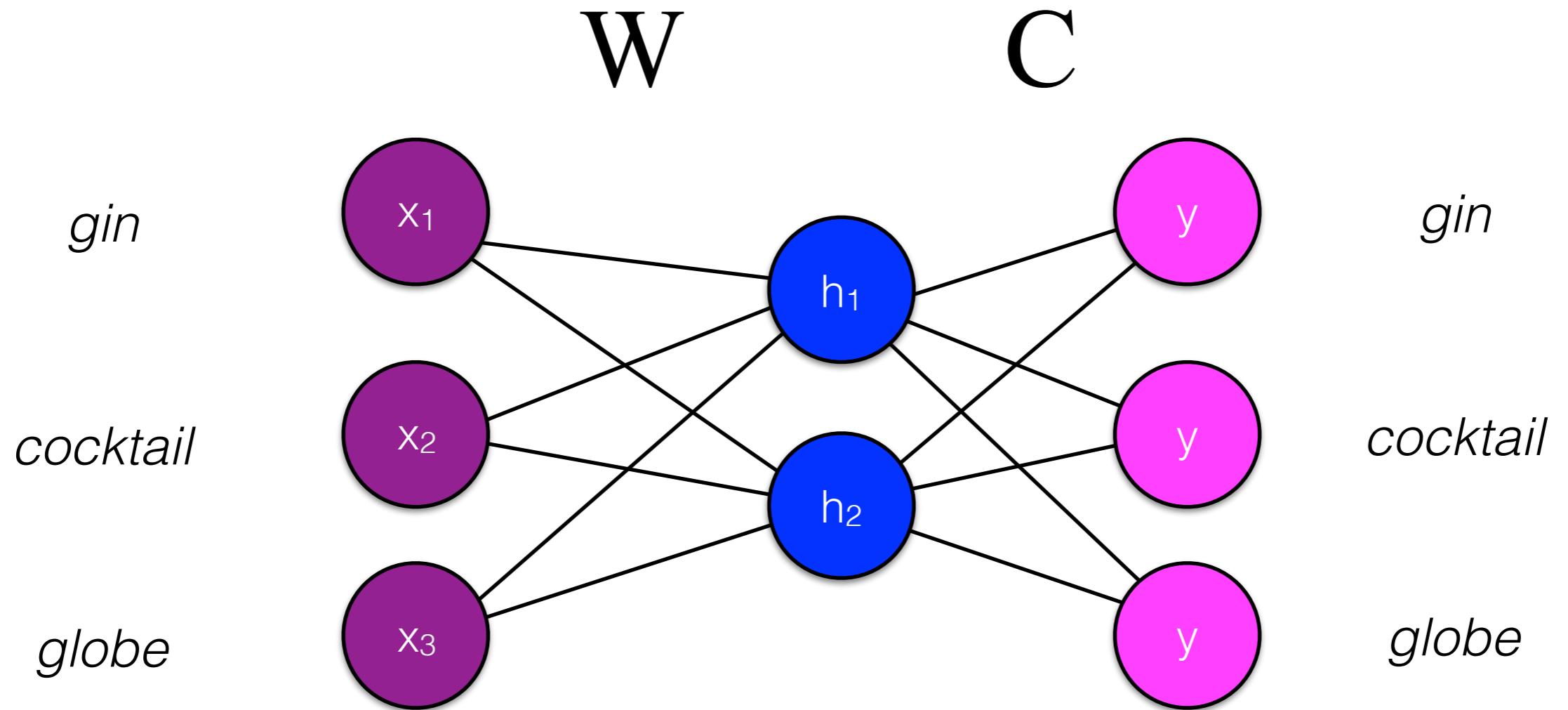
y  
1  
0  
0



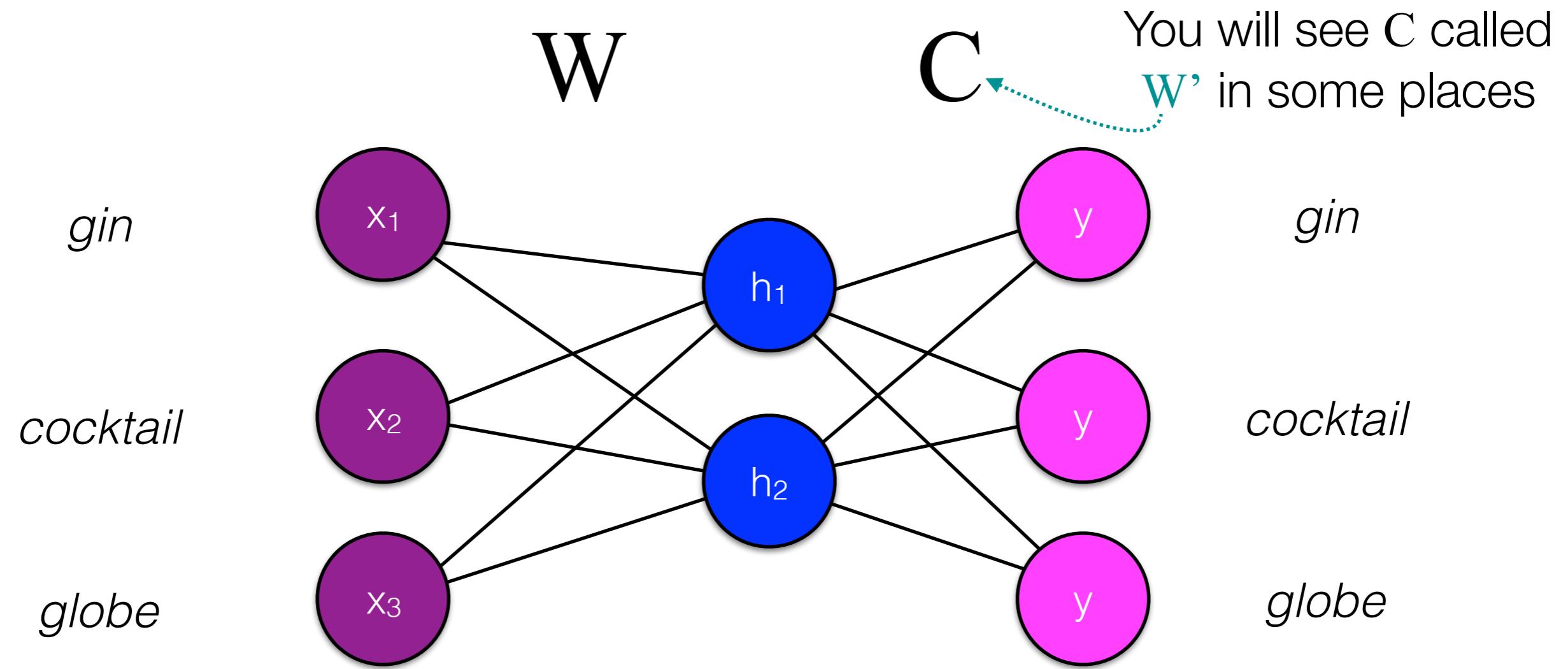
	x	W		y
gin	0	-0.5	1.3	1
cocktail	1	0.4	0.08	0
globe	0	1.7	3.1	0



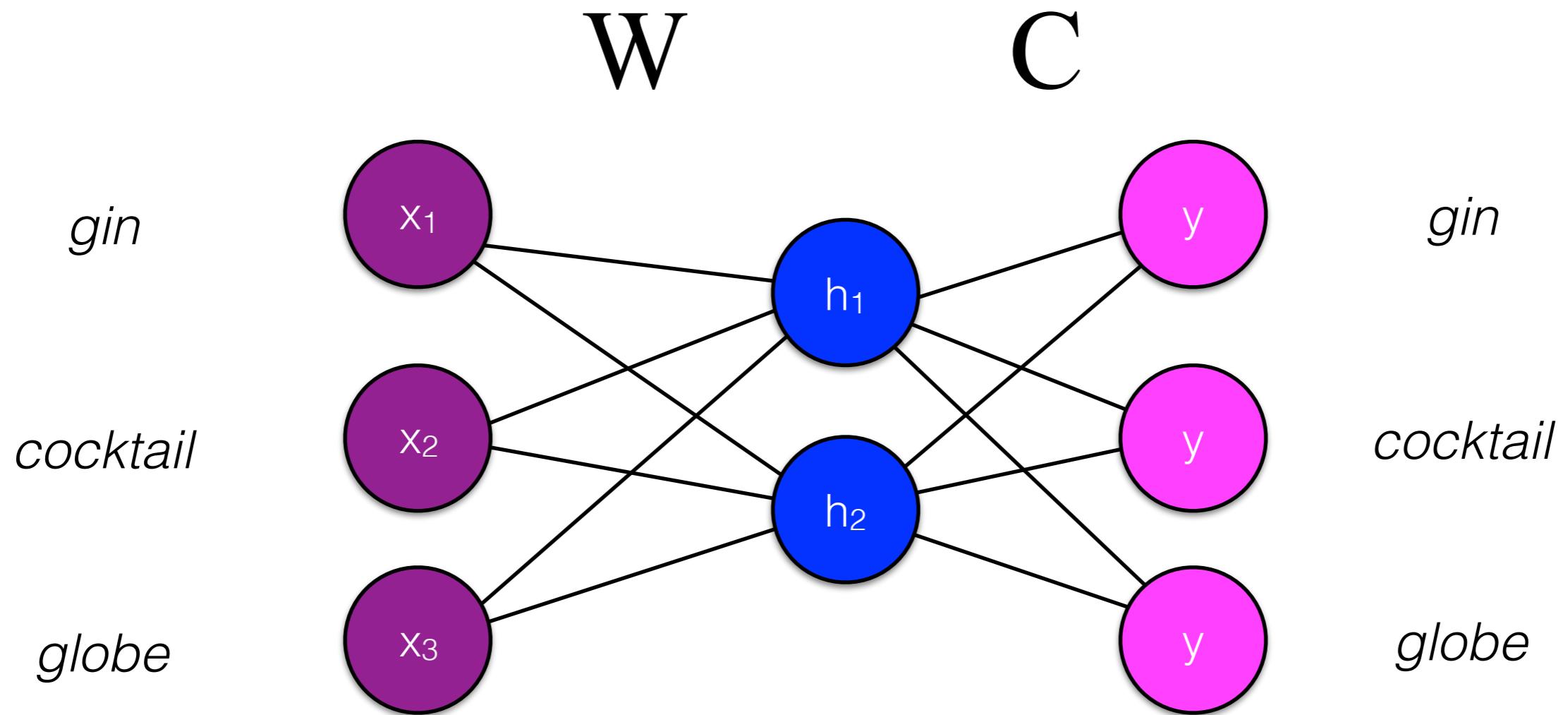
	x	W		y
gin	0	-0.5	1.3	1
cocktail	1	0.4	0.08	0
globe	0	1.7	3.1	0



	x	w	c	y
gin	0	-0.5 1.3	4.1 0.7 0.1	1
cocktail	1	0.4 0.08	-0.9 1.3 0.3	0
globe	0	1.7 3.1		0



x	W		C			y	
<i>gin</i>	0	-0.5	1.3	4.1	0.7	0.1	1
<i>cocktail</i>	1	0.4	0.08	-0.9	1.3	0.3	0
<i>globe</i>	0	1.7	3.1				0



Only one of the inputs is nonzero.

= the inputs are really the  $W$  matrix

W	
-0.5	1.3
0.4	0.08
1.7	3.1

C		
4.1	0.7	0.1
-0.9	1.3	0.3

X

W

1

0.13 0.56

-1.75 0.07

0.80 1.19

-0.11 1.38

-0.62 -1.46

-1.16 -1.24

0.99 -0.26

-1.46 -0.85

0.79 0.47

0.06 -1.21

-0.31 0.00

1

-1.01 -2.52

$x^\top W =$

-1.01 -2.52

1

-1.50 -0.14

-0.14 0.01

-0.13 -1.76

-1.08 -0.56

-0.17 -0.74

0.31 1.03

-0.24 -0.84

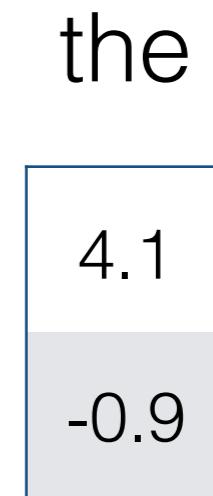
-0.79 -0.18

This is the  
embedding of the  
context

# Dimensionality reduction = Projection = Embedding

...	...
the	1
a	0
an	0
for	0
in	0
on	0
dog	0
cat	0
...	...

*the* is a point in V-dimensional space



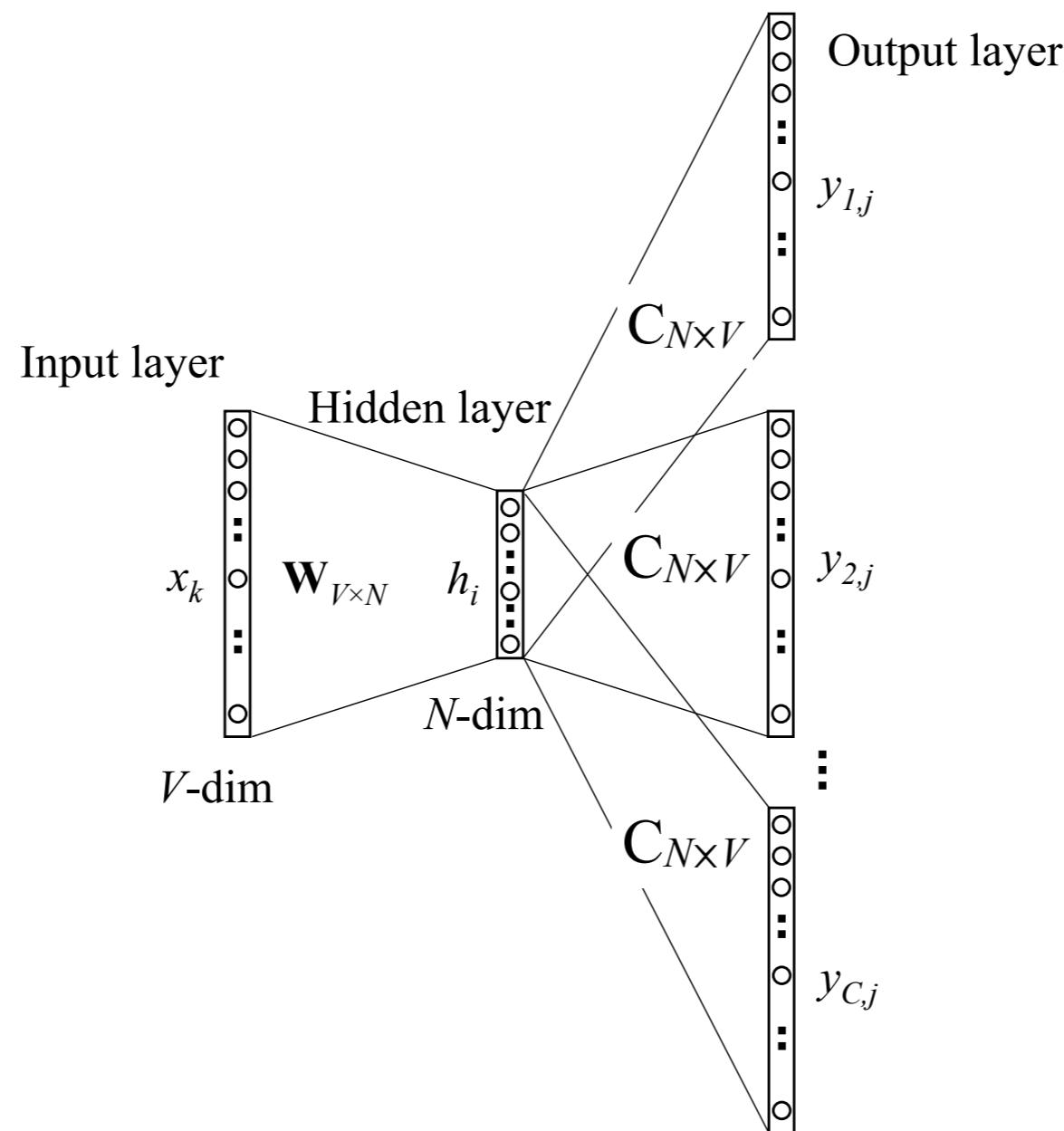
*the* is a point in 2-dimensional space

# Word embeddings

- Can you predict the output word from a **vector representation** of the input word?
- Rather than seeing the input as a one-hot encoded vector specifying the word in the vocabulary we're conditioning on, we can see it as **indexing** into the appropriate row in the weight matrix  $W$

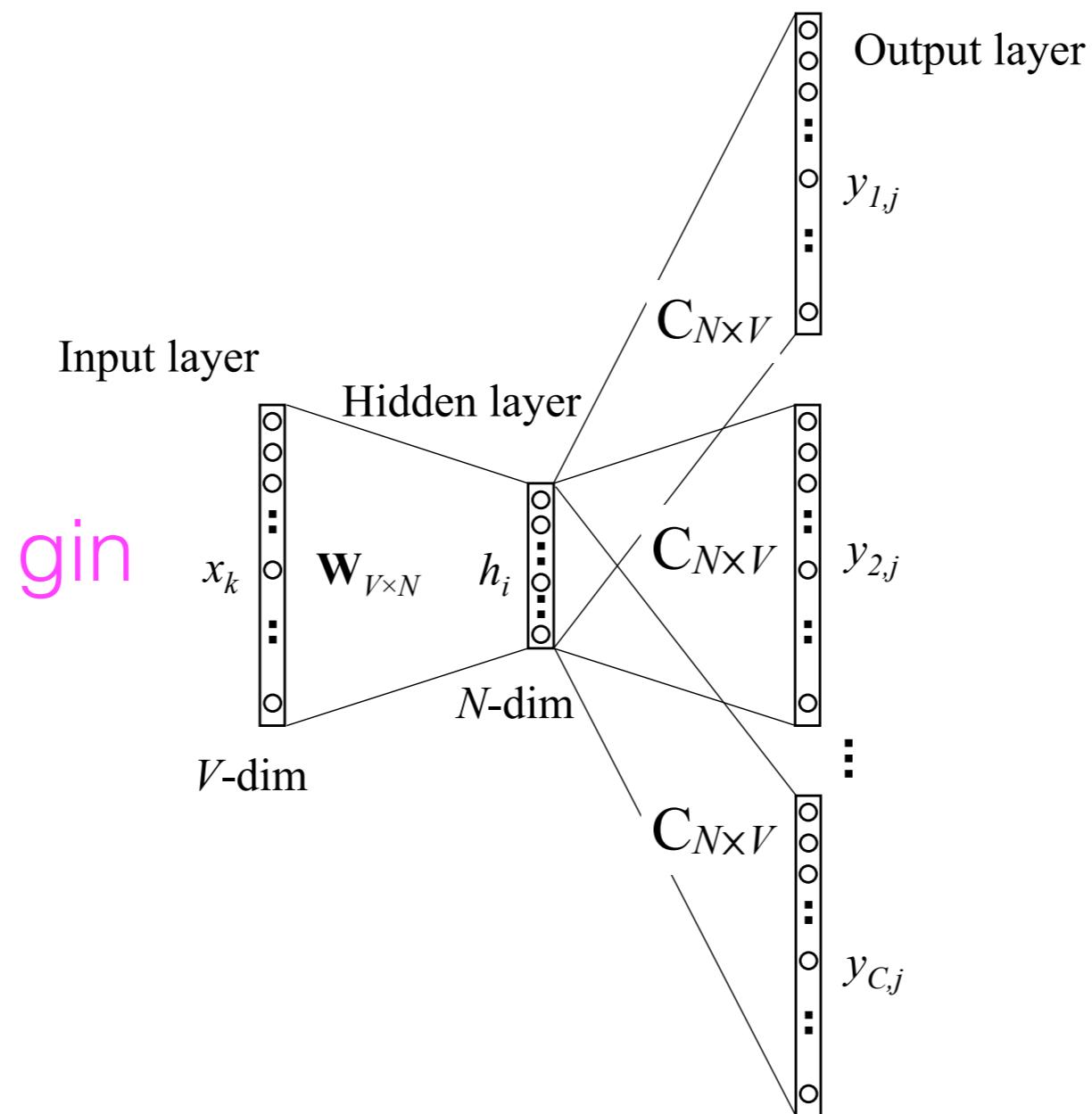
# word2vec: Predicting the likely context for a word

**Training example:** a cocktail with **gin** and seltzer



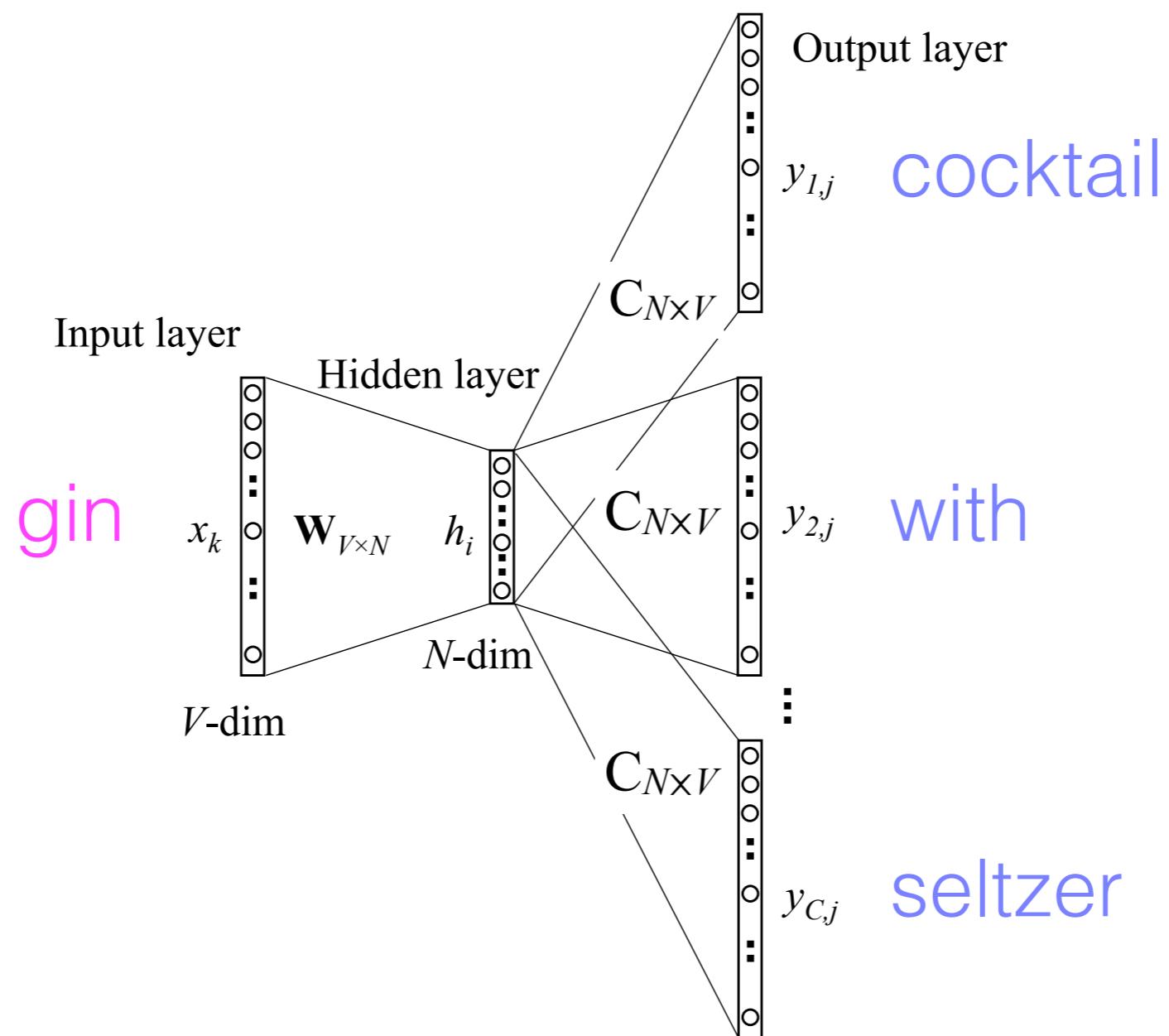
# word2vec: Predicting the likely context for a word

**Training example:** a cocktail with **gin** and seltzer



# word2vec: Predicting the likely context for a word

**Training example:** a cocktail with **gin** and seltzer



# Setup

# Setup

Let's represent words as vectors of some length (say 300), randomly initialized.

# Setup

Let's represent words as vectors of some length (say 300), randomly initialized.

So we start with  $300 * V$  random parameters

# Setup

Let's represent words as vectors of some length (say 300), randomly initialized.

So we start with  $300 * V$  random parameters

Over the entire training set, we'd like to adjust those word vectors such that we

Maximize the similarity of the **target word**, **context word** pairs  $(t, c)$  drawn from the positive data

Minimize the similarity of the  $(t, c_n)$  pairs drawn from the **negative data**.

# Learning the classifier

Iterative process.

We'll start with 0 or random weights

Then adjust the word weights to  
make the positive pairs more likely  
and the negative pairs less likely

over the entire training set:

# Objective Criteria

We want to maximize...

$$\sum_{(t,c) \in +} \log P(+) | t, c) + \sum_{(t,c) \in -} \log P(- | t, c)$$

Maximize the + label for the pairs from the **positive training data**, and the – label for the **negative pairs sampled from the data**.

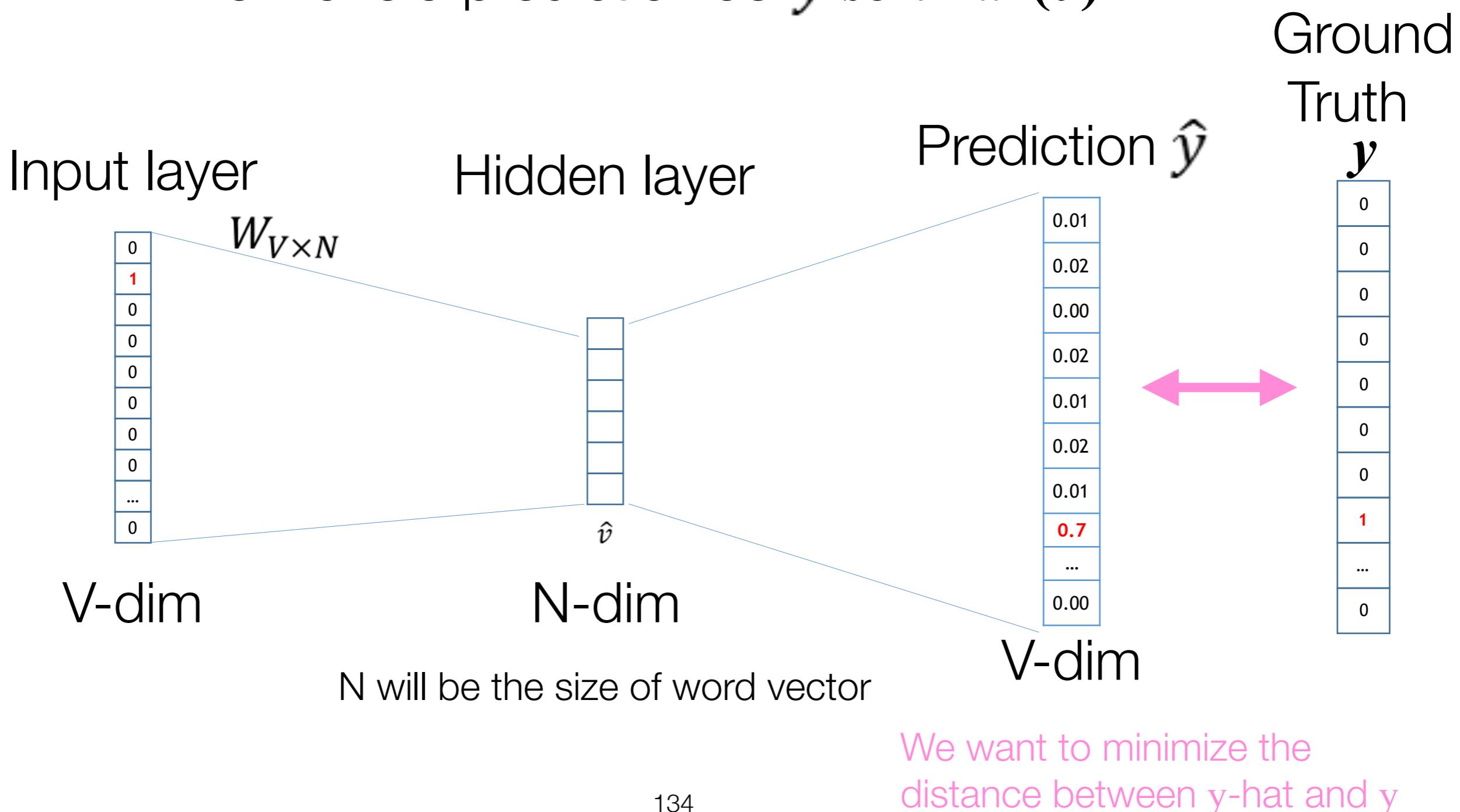
# Train using gradient descent

Actually learns two separate embedding matrices  $W$  and  $C$

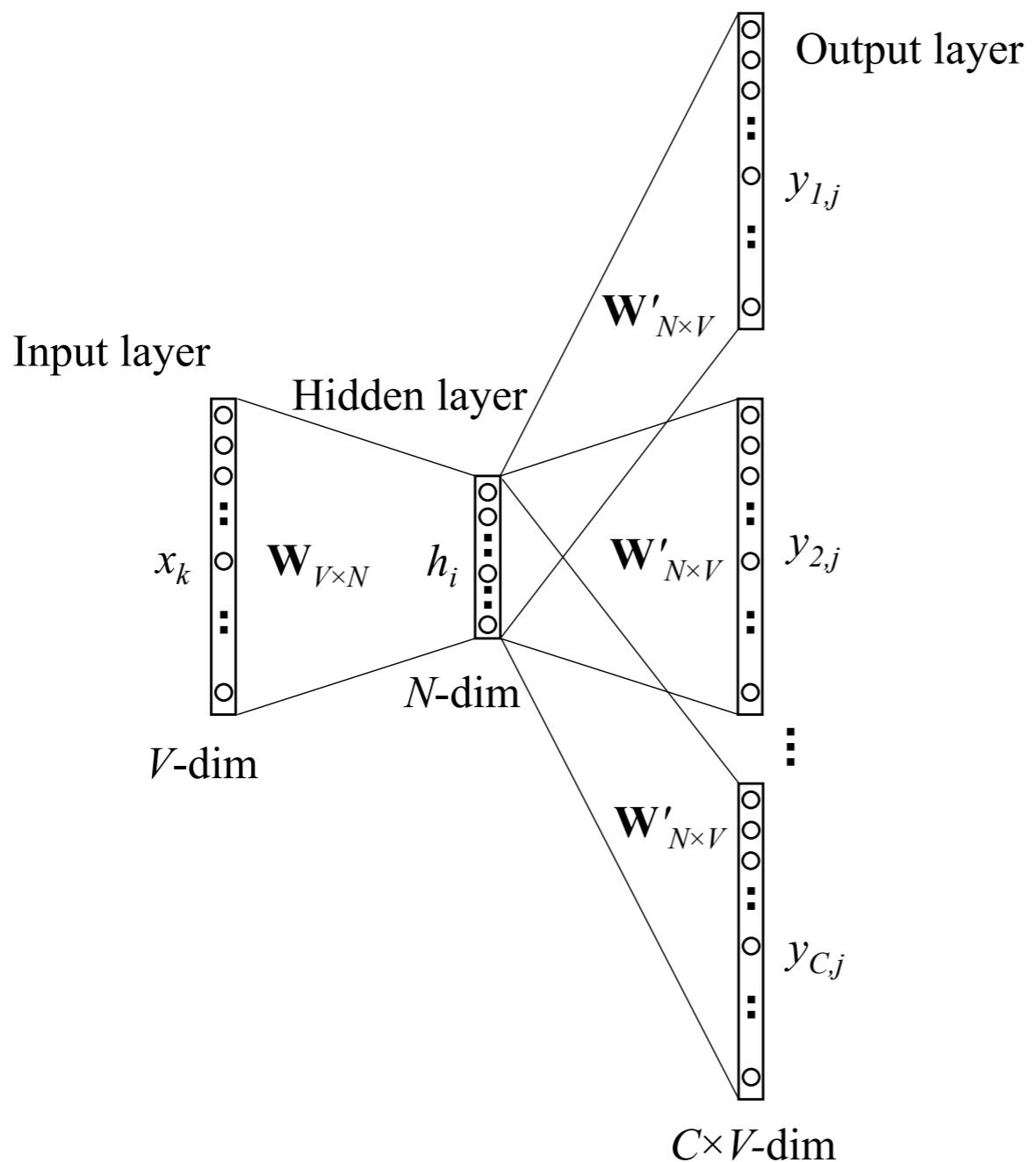
Can use  $W$  and throw away  $C$ , or merge them somehow

# Calculating Error

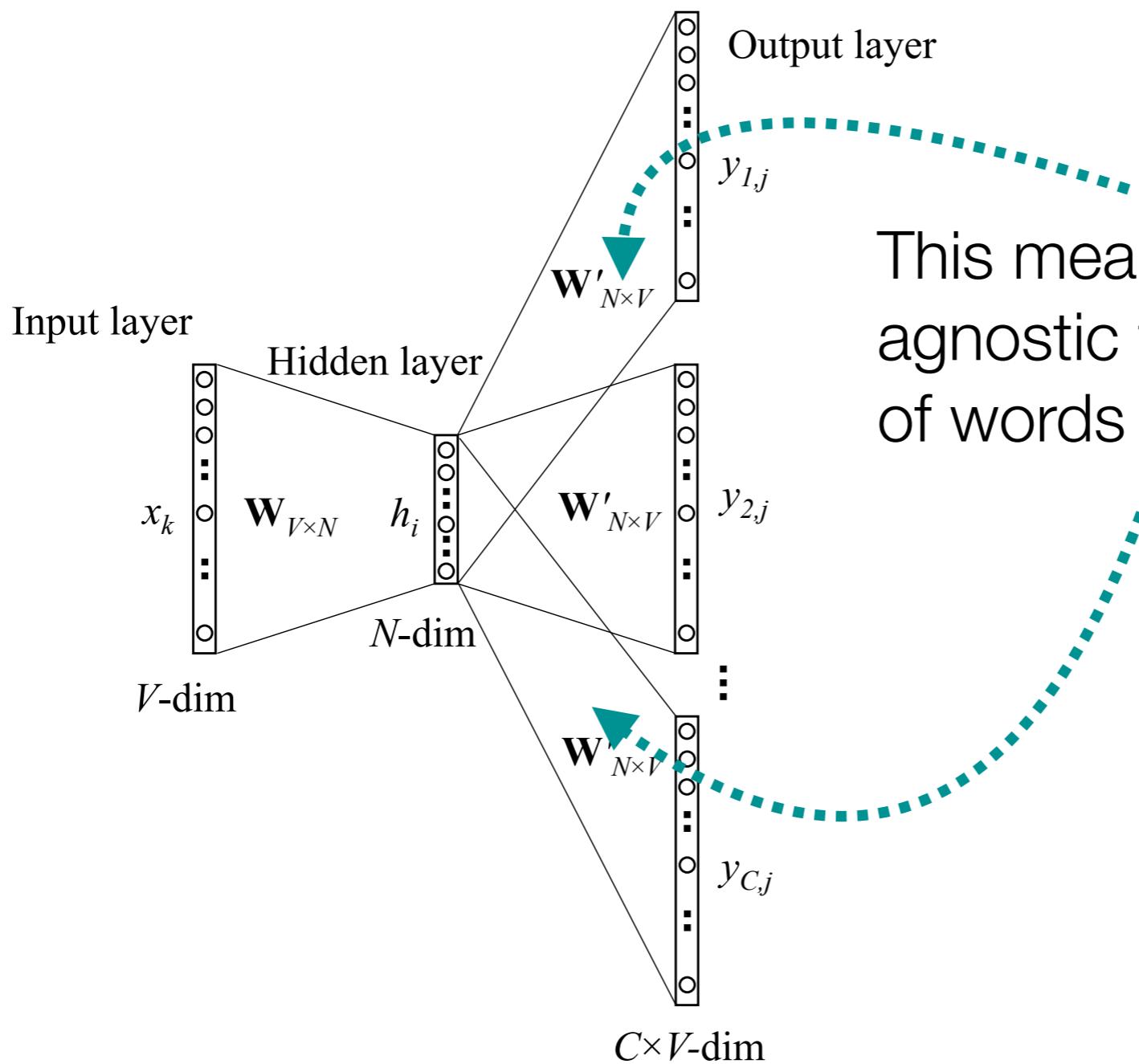
Given the output layer's activation  $\mathbf{u} = \mathbf{W}'\mathbf{h}$   
we make a prediction as  $\hat{\mathbf{y}} \text{ softmax}(\mathbf{u})$



Import note: All our predictions share the same  $\mathbf{W}'$  matrix!

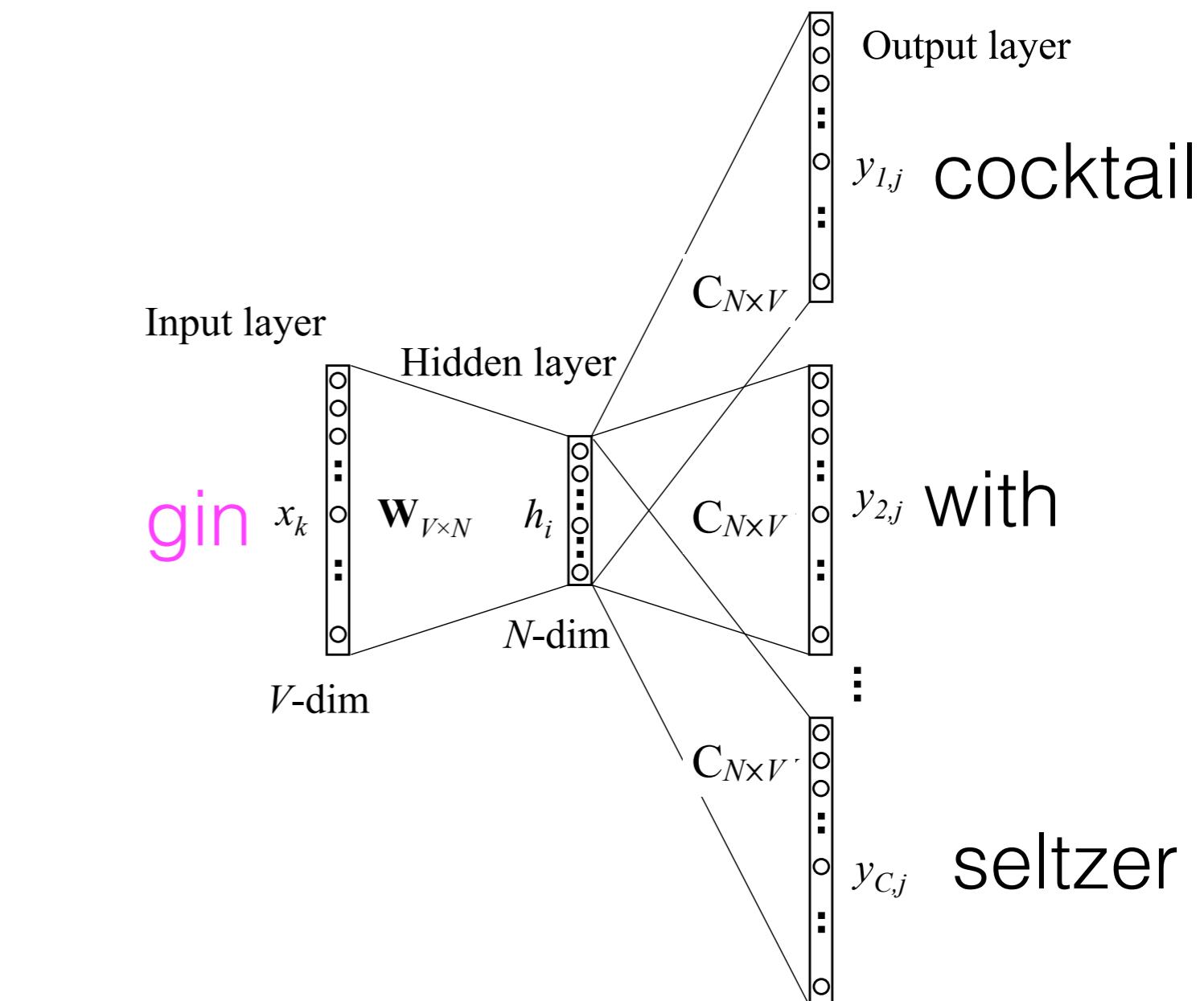


# Import note: All our predictions share the same $\mathbf{W}'$ matrix!



This means word2vec is agnostic to the relative position of words in the context.

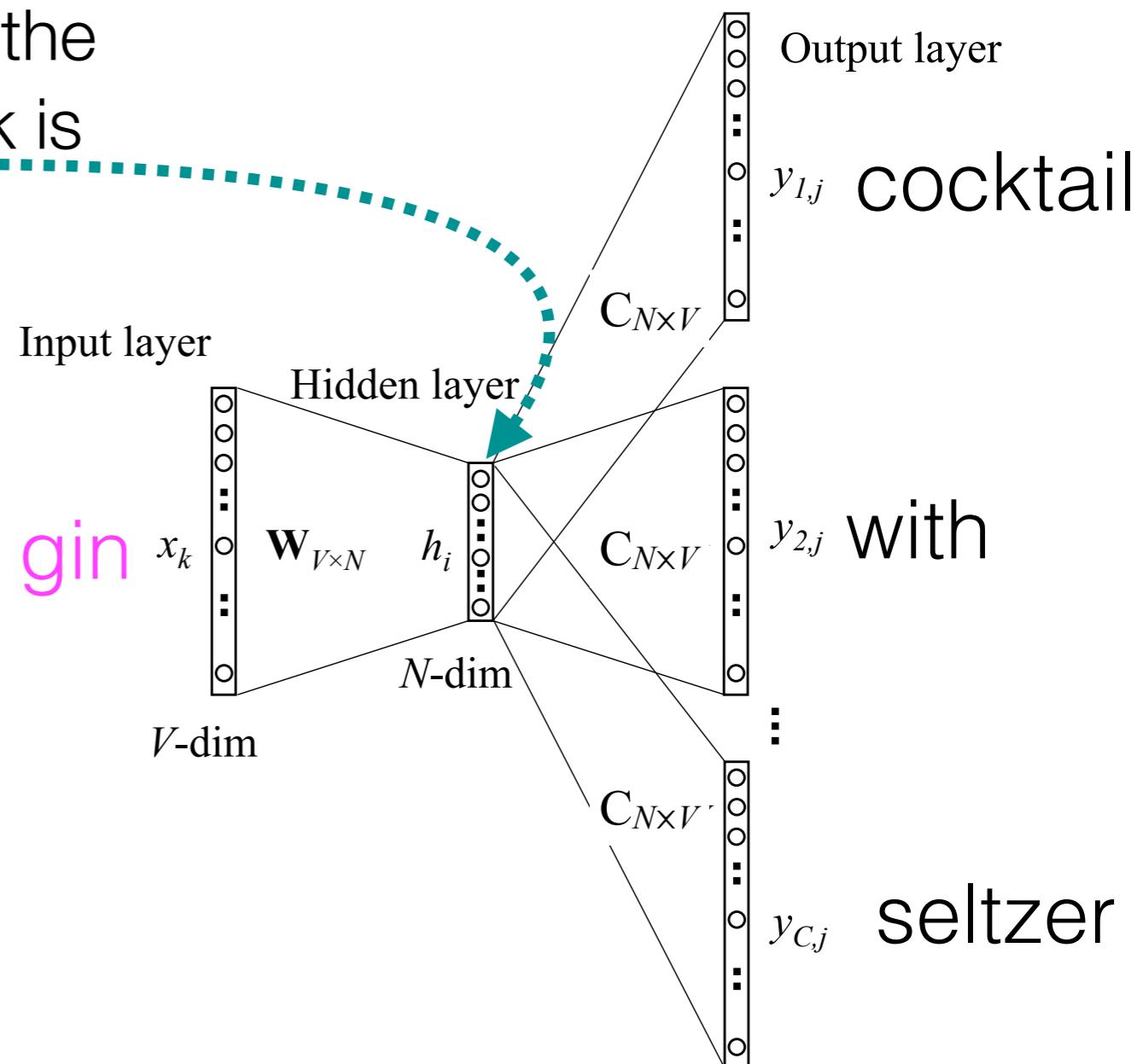
# Training word2vec: making predictions



# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the hidden layer activation in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

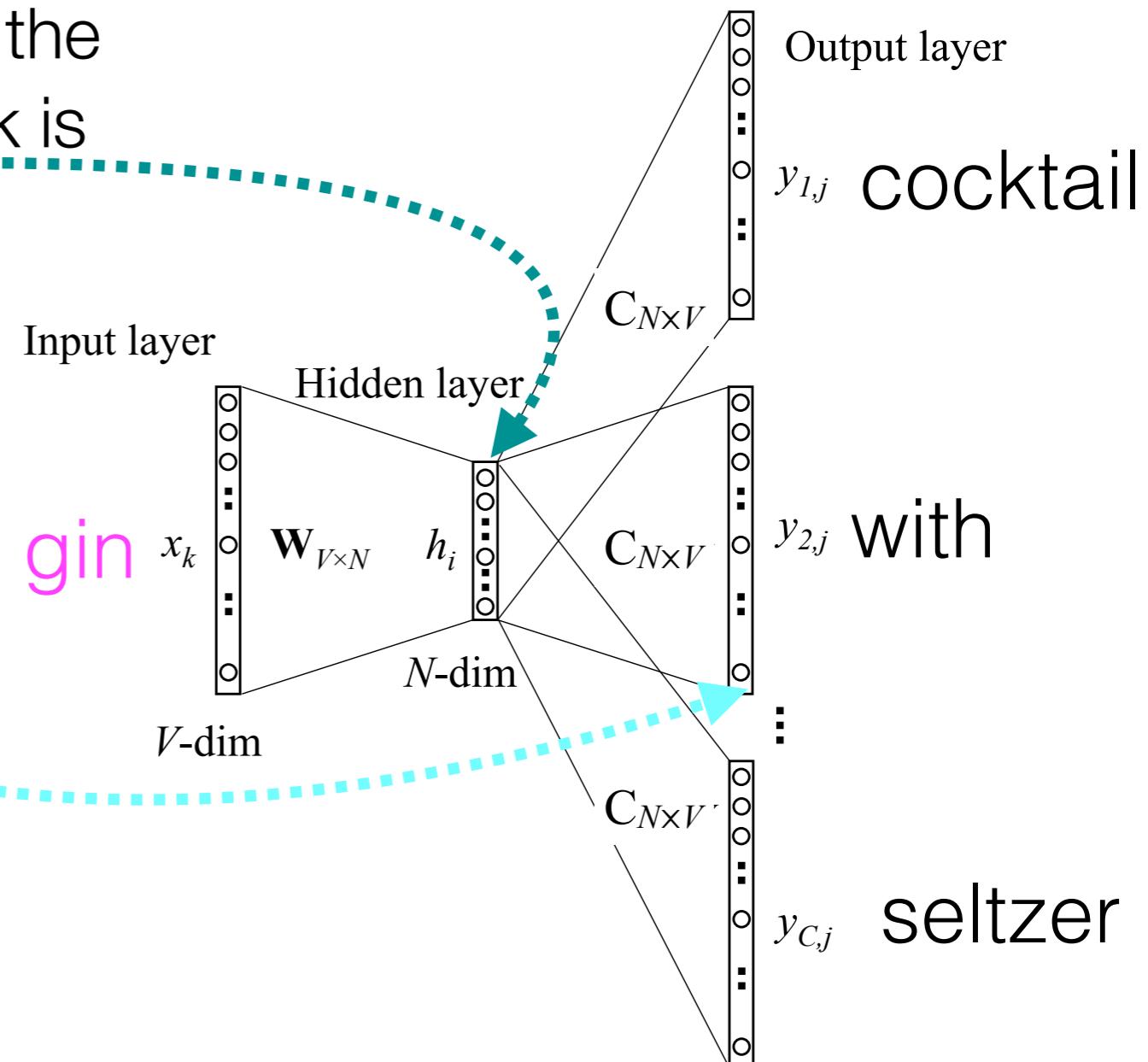


# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the **hidden layer activation** in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j$ :  $\mathbf{u}_j = \mathbf{W}' \mathbf{h}$



# Training word2vec: making predictions

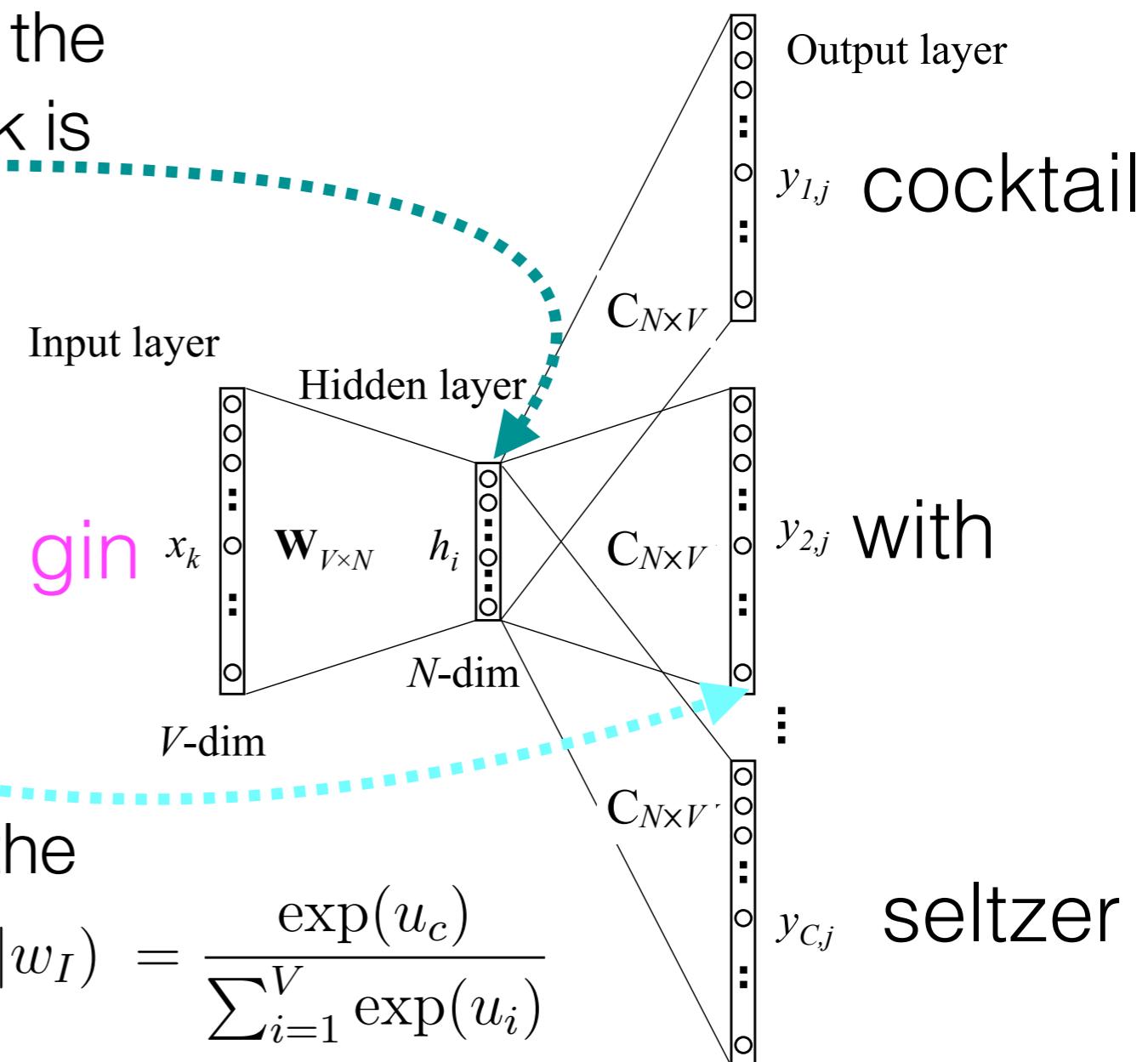
Given the input vector  $\mathbf{x}$  for word  $w_i$ , the **hidden layer activation** in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j$ :  $\mathbf{u}_j = \mathbf{W}' \mathbf{h}$

We predict the probability of  $w_j$  from the output layer's activation as

$$P(w_c = w_c^* | w_I) = \frac{\exp(u_c)}{\sum_{i=1}^V \exp(u_i)}$$



# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the hidden layer activation in the network is

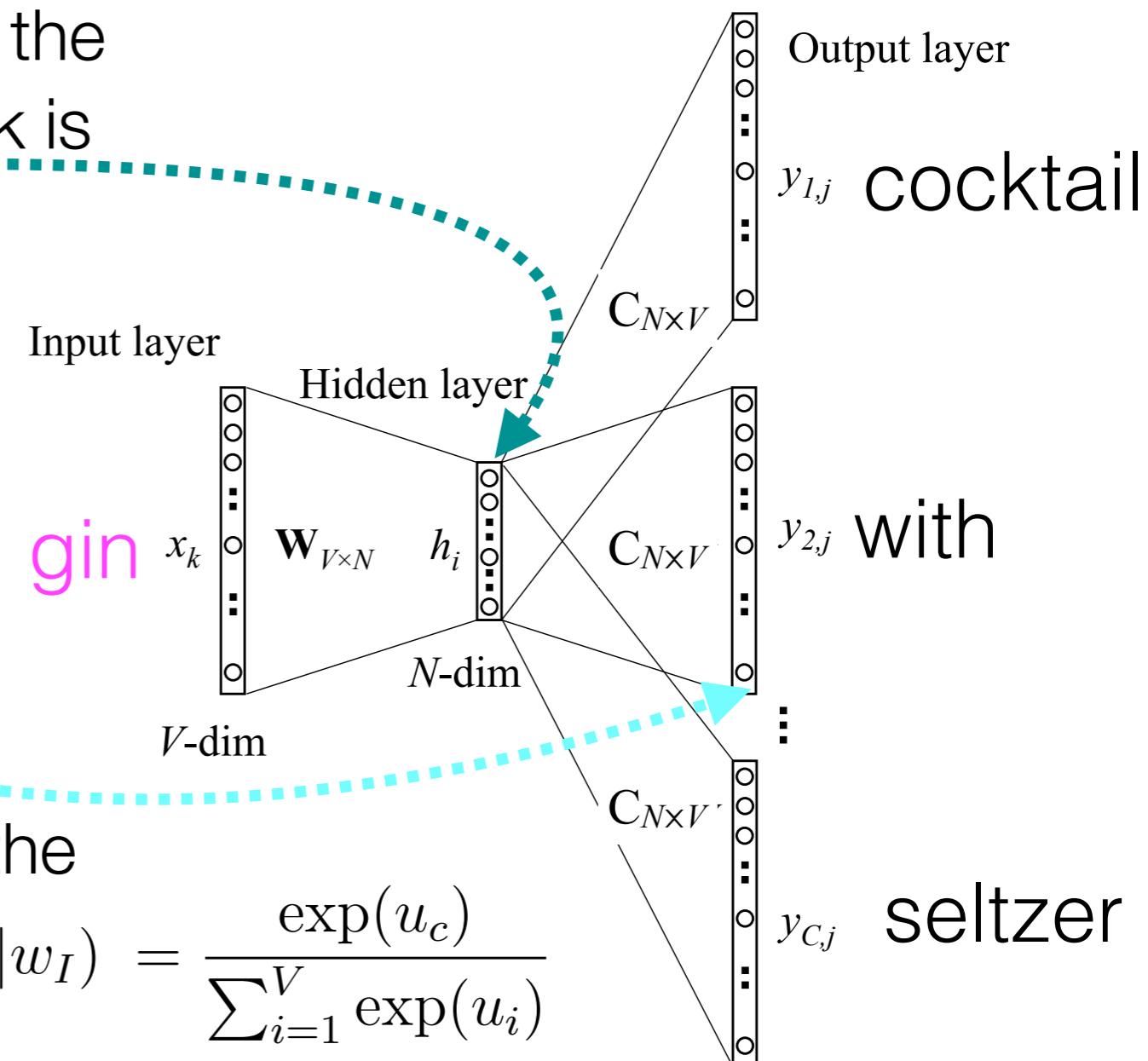
$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j$ :  $\mathbf{u}_j = \mathbf{W}' \mathbf{h}$

We predict the probability of  $w_j$  from the output layer's activation as

$$P(w_c = w_c^* | w_I) = \frac{\exp(u_c)}{\sum_{i=1}^V \exp(u_i)}$$

$w_c$  is the predicted context word



# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the hidden layer activation in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

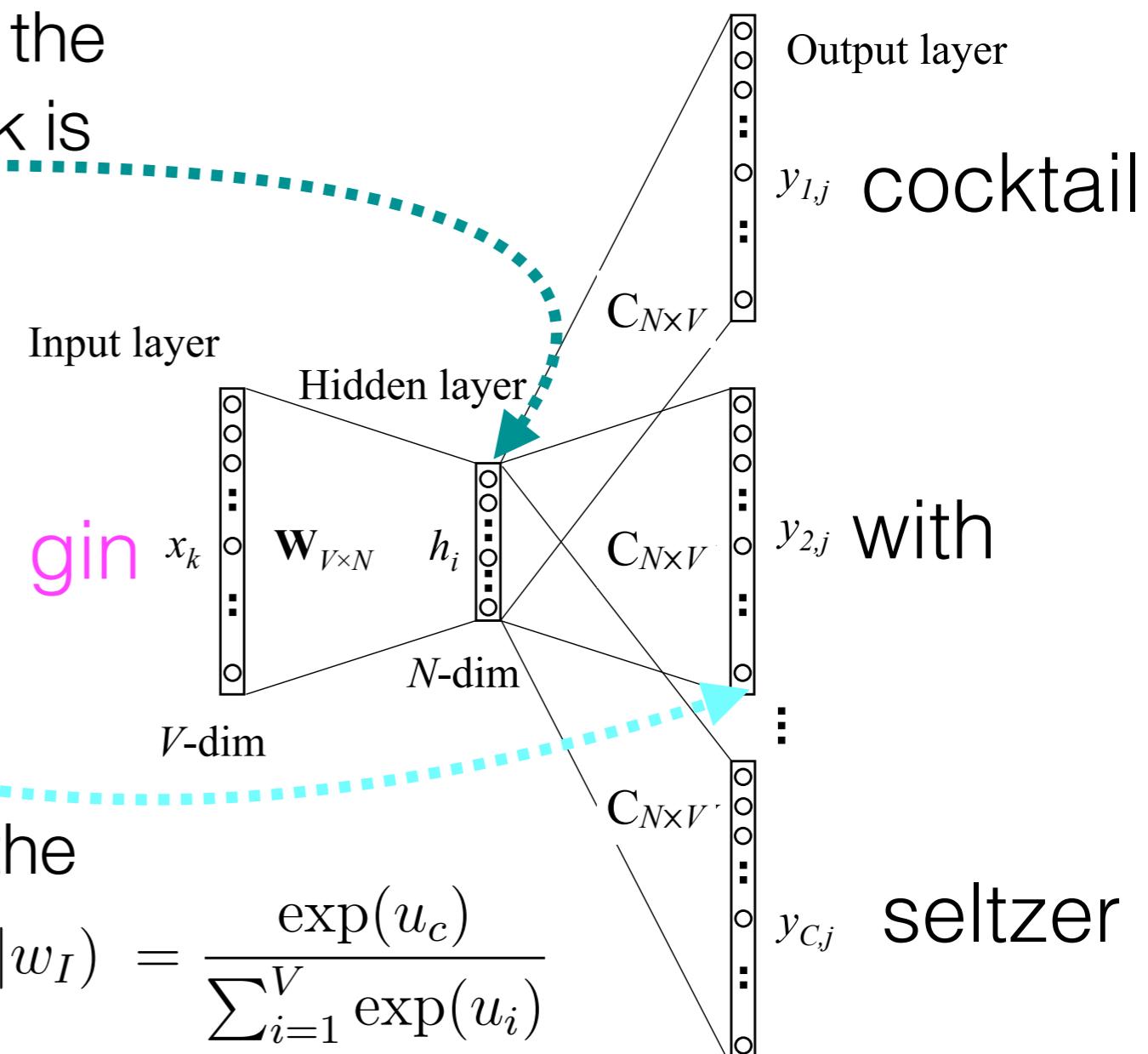
We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j = \mathbf{W}'\mathbf{h}$

We predict the probability of  $w_j$  from the output layer's activation as

$$P(w_c = w_c^* | w_I) = \frac{\exp(u_c)}{\sum_{i=1}^V \exp(u_i)}$$

$w_c$  is the predicted context word

$w_c^*$  is the actual context word



# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the hidden layer activation in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j = \mathbf{W}'\mathbf{h}$

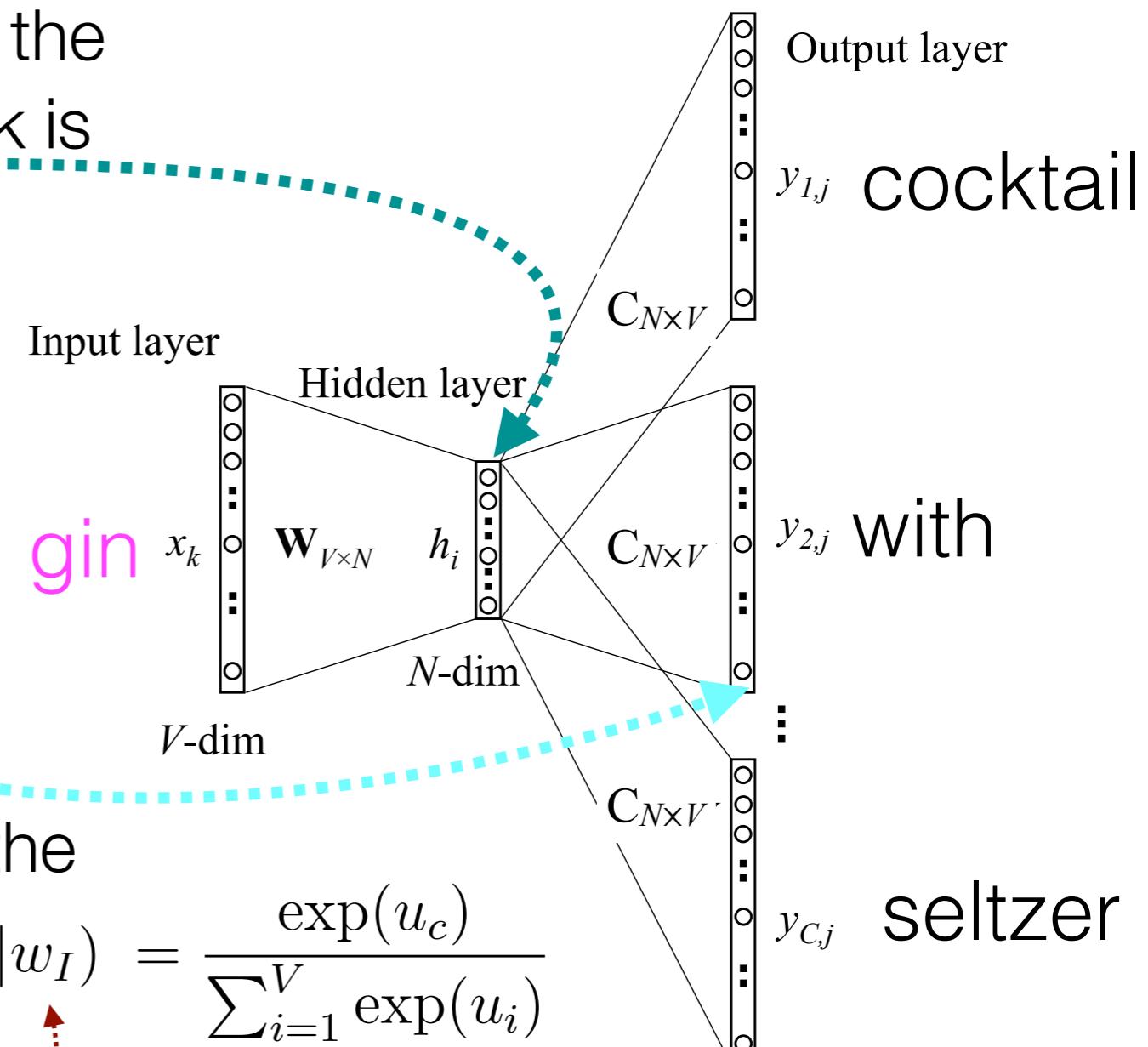
We predict the probability of  $w_j$  from the output layer's activation as

$$P(w_c = w_c^* | w_I) = \frac{\exp(u_c)}{\sum_{i=1}^V \exp(u_i)}$$

$w_c$  is the predicted context word

$w_c^*$  is the actual context word

$w_I$  is the word we saw as input



# Training word2vec: making predictions

Given the input vector  $\mathbf{x}$  for word  $w_i$ , the hidden layer activation in the network is

$$\mathbf{h} = \mathbf{W}^T \mathbf{x}$$

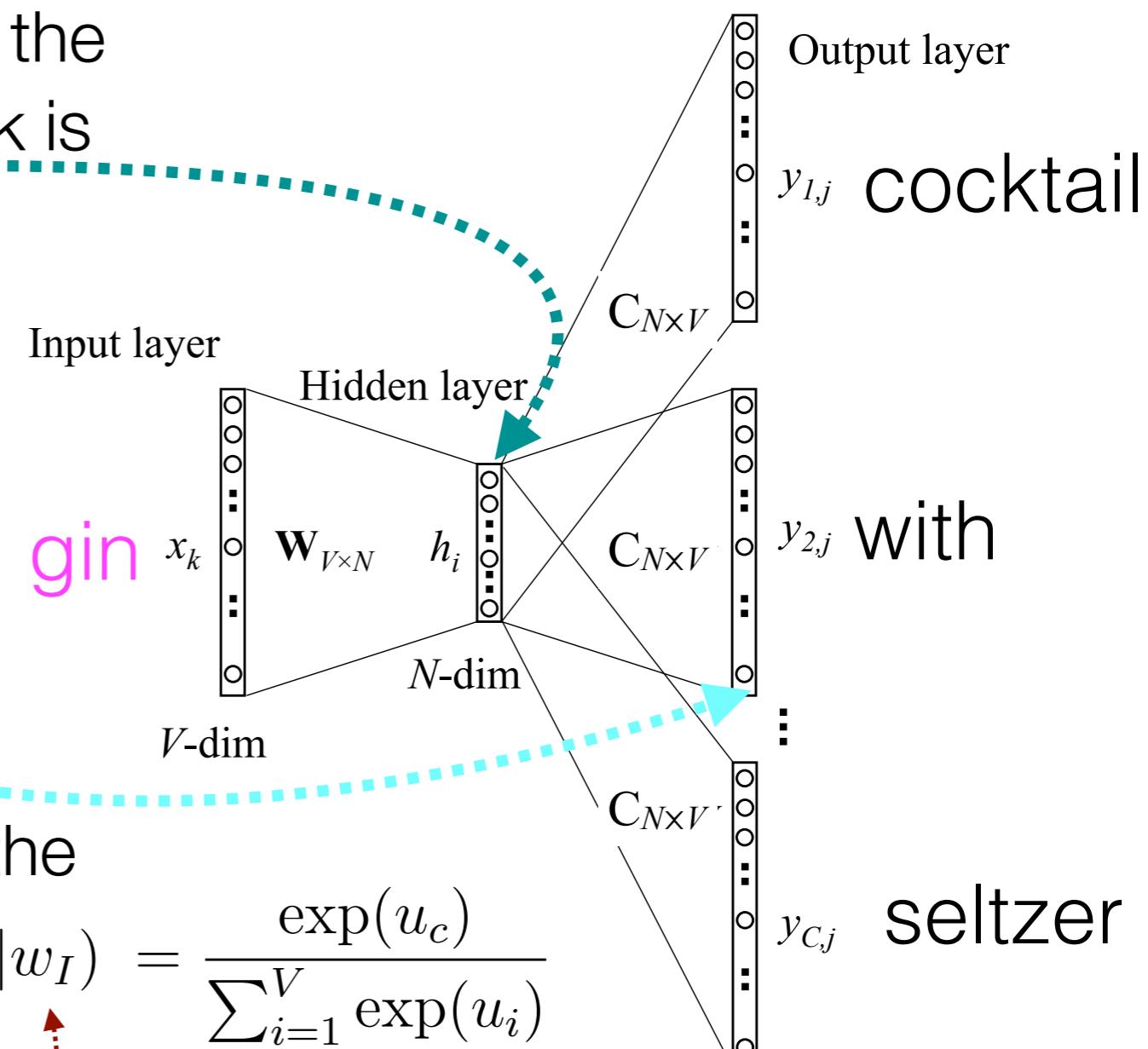
We produce the output layer's activation of context word  $w_j$  by multiplying  $\mathbf{h}$  with the vector for  $w_j$  in  $\mathbf{W}'$ , which we'll call  $\mathbf{u}_j = \mathbf{W}'\mathbf{h}$

We predict the probability of  $w_j$  from the output layer's activation as

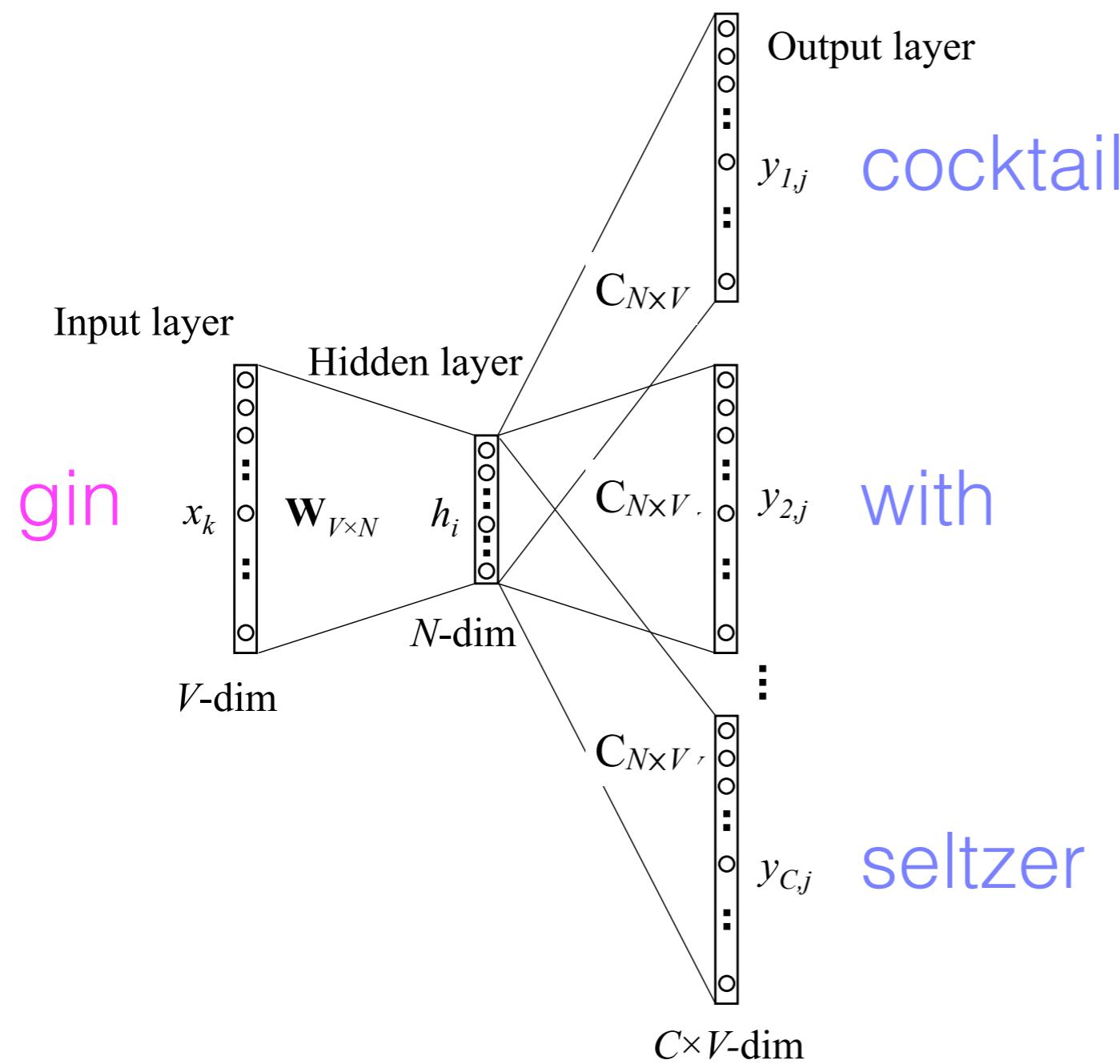
$w_c^*$  is the predicted context word

$w_c^*$  is the actual context word

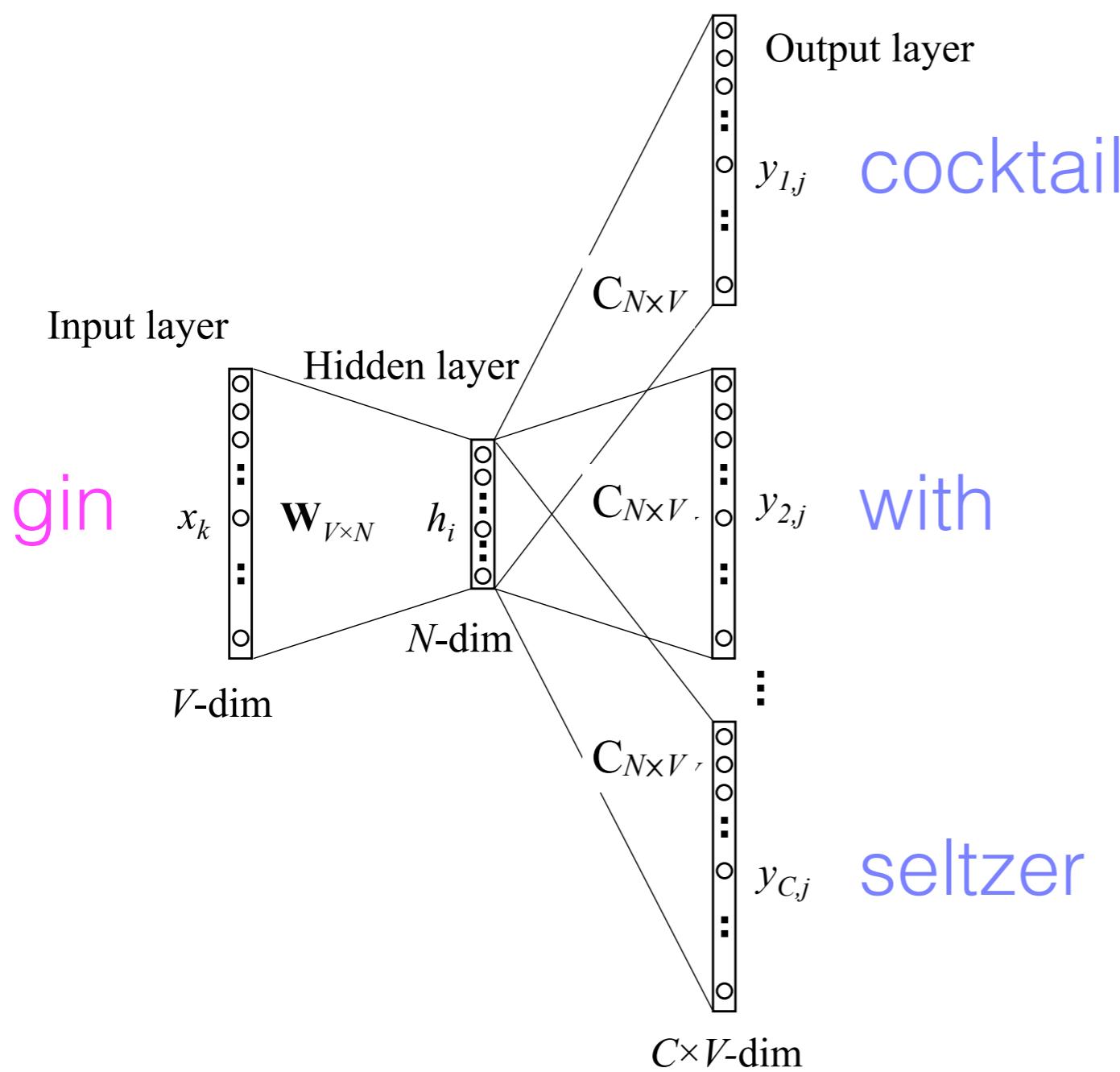
$w_I$  is the word we saw as input



**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



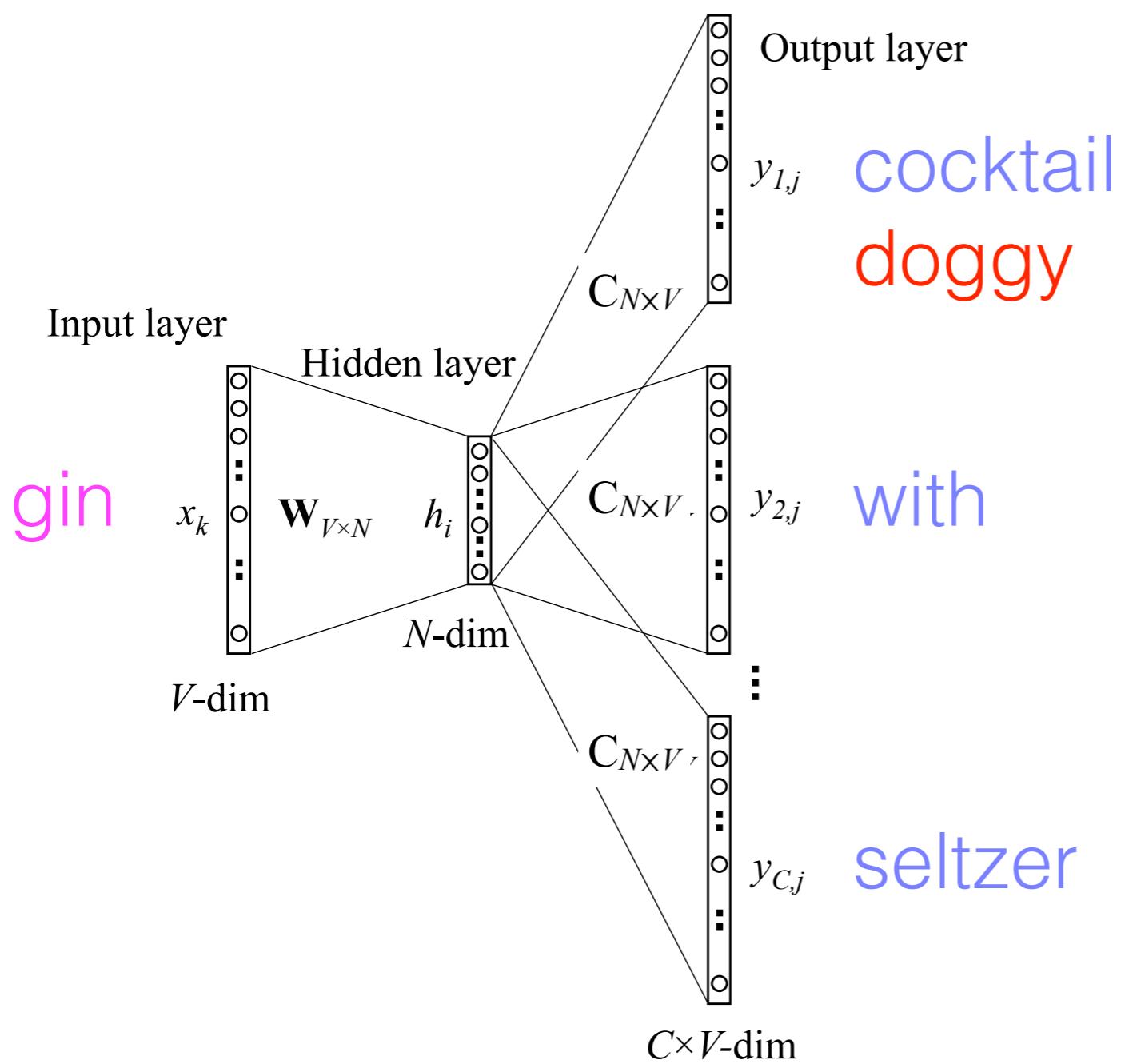
**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



Let  $f(w_i)$  be the frequency of a word occurring in our corpus; choose that word with probability:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

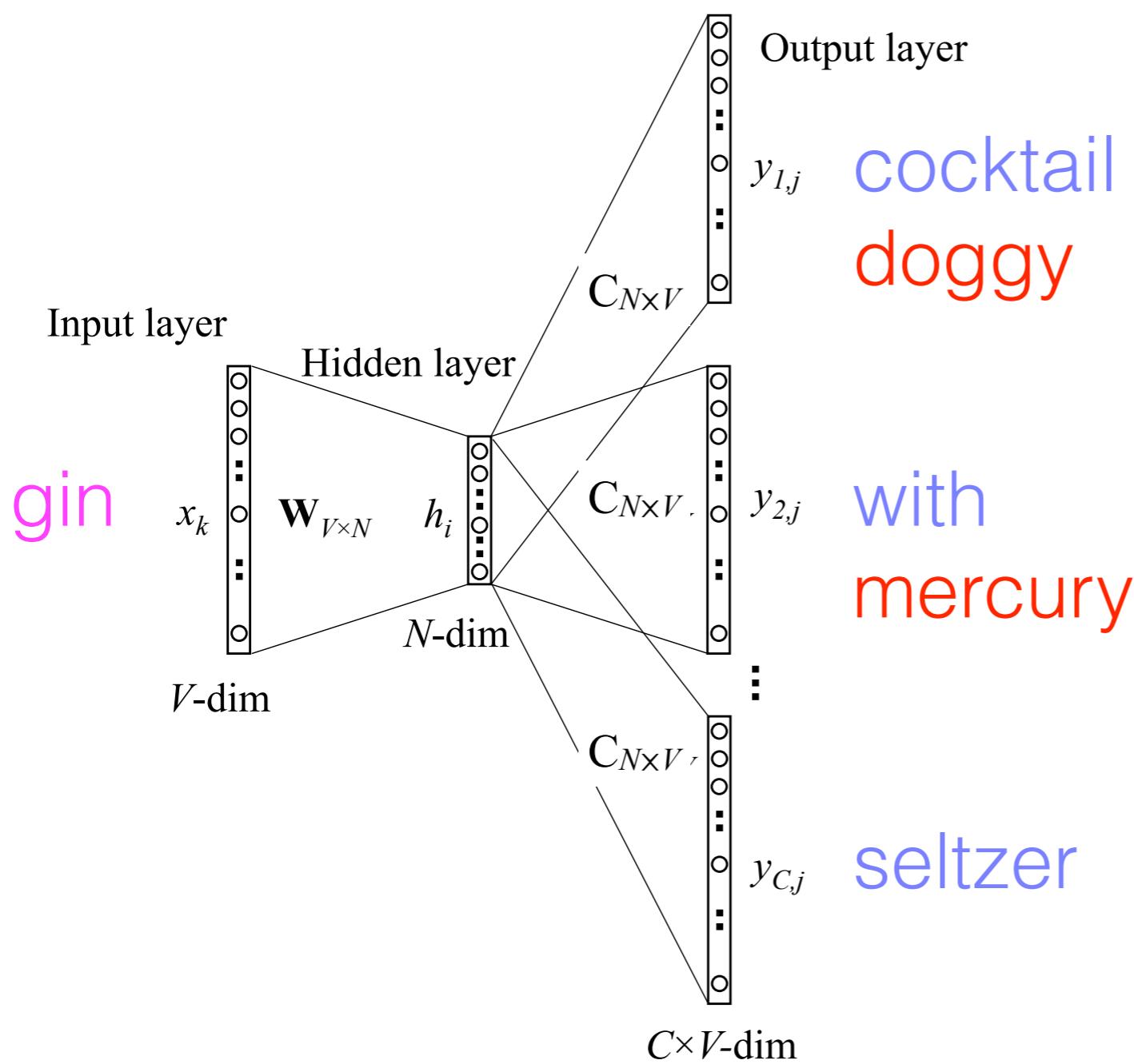
**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



Let  $f(w_i)$  be the frequency of a word occurring in our corpus; choose that word with probability:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

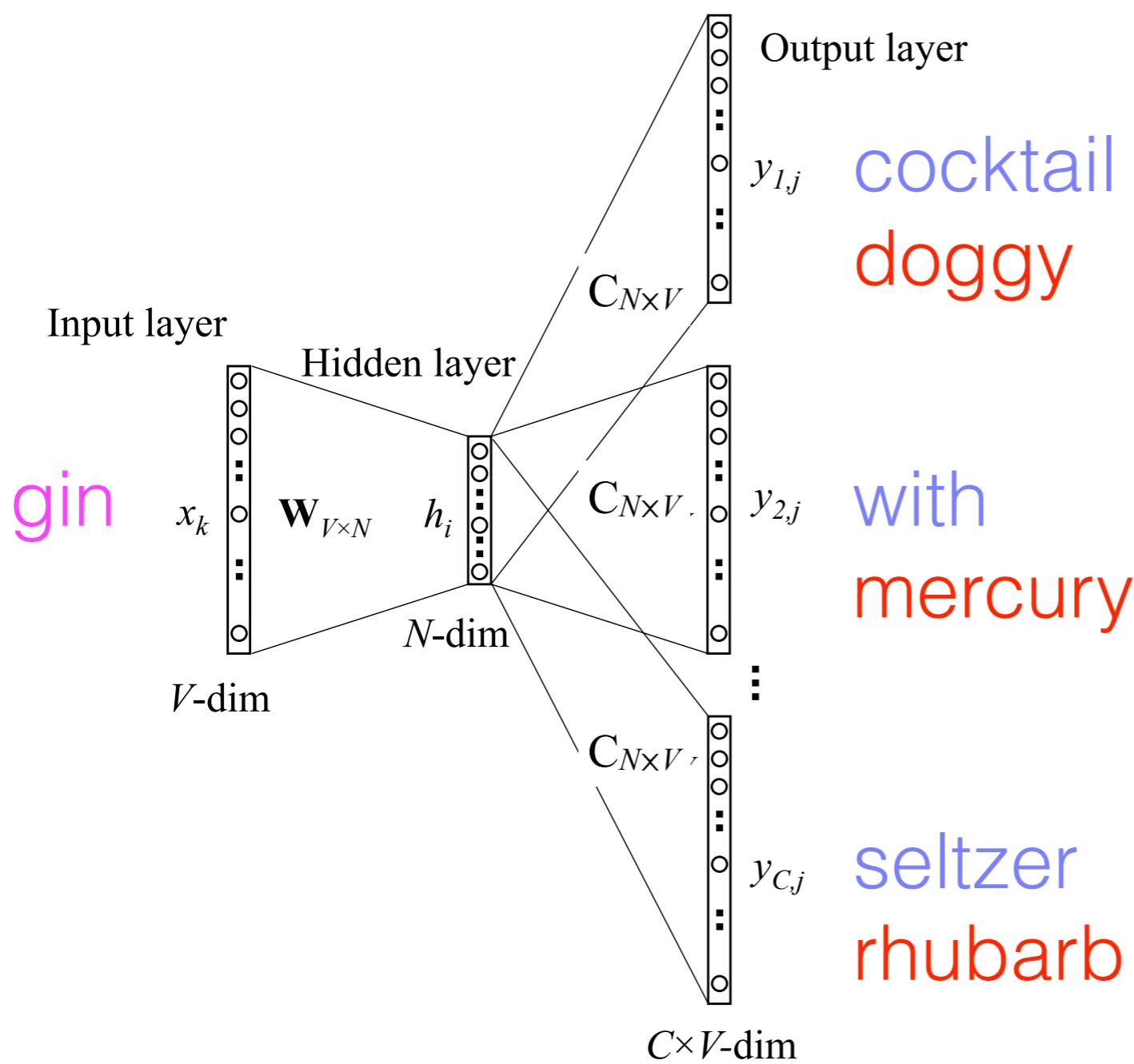
**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



Let  $f(w_i)$  be the frequency of a word occurring in our corpus; choose that word with probability:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

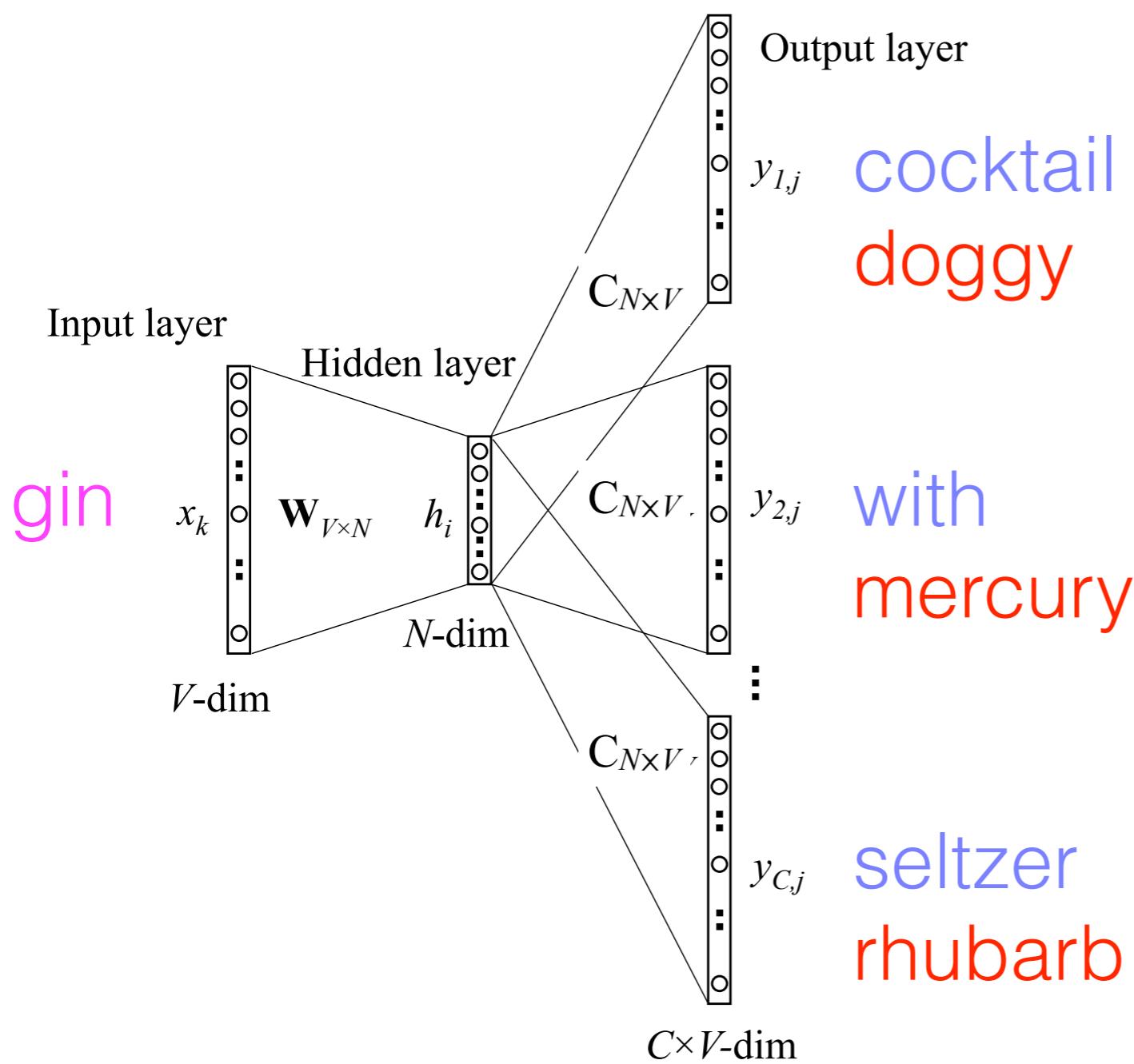
**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



Let  $f(w_i)$  be the frequency of a word occurring in our corpus; choose that word with probability:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

**Negative Sampling:** Sample a **small number** of words according to our corpus distribution as **negative examples**, choosing rare words more frequently



Let  $f(w_i)$  be the frequency of a word occurring in our corpus; choose that word with probability:

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^n (f(w_j)^{3/4})}$$

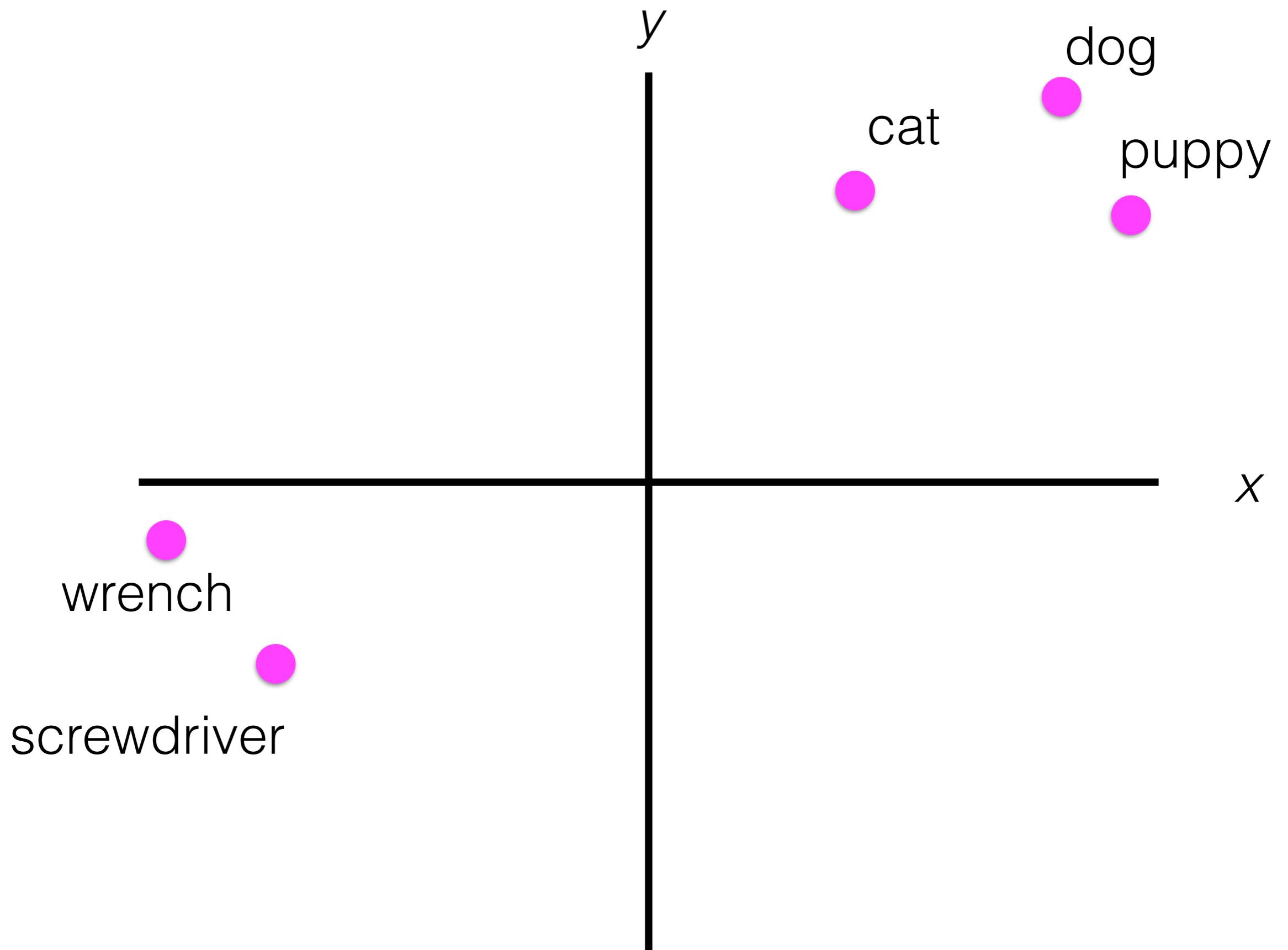
**In practice, people choose 5-10 negative examples**

# Word embeddings

- Similarly, C has one H-dimensional vector for each element in the vocabulary (for the words that are being predicted)

C			
gin	cocktail	cat	globe
4.1	0.7	0.1	1.3
-0.9	1.3	0.3	-3.4

This is also an embedding of the word



- Why this behavior? *dog*, *cat* show up in similar positions

the	black	[0.4, 0.08]	jumped	on	the	table
the	black	[0.4, 0.07]	jumped	on	the	table
the	black	puppy	jumped	on	the	table
the	black	skunk	jumped	on	the	table
the	black	shoe	jumped	on	the	table

- Why this behavior? *dog*, *cat* show up in similar positions

the	black	[0.4, 0.08]	jumped	on	the	table
the	black	[0.4, 0.07]	jumped	on	the	table
the	black	puppy	jumped	on	the	table
the	black	skunk	jumped	on	the	table
the	black	shoe	jumped	on	the	table

To make the same predictions, these numbers need to be close to each other.

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

Start with W and C matrices as random 300-dimensional vectors as initial embeddings for each word

Use a one layer neural network, with weight matrices W and C

1. Take a corpus and take pairs of words that co-occur as **positive examples**
2. Take pairs of words that are unlikely to co-occur as **negative examples**
3. Train the network to distinguish these by slowly adjusting all the embeddings in W and C to improve the network performance
4. Throw away the network code and keep the embeddings.



# Evaluation

# Evaluating embeddings

Compare to human scores on word similarity-type tasks:

WordSim-353 (Finkelstein et al., 2002)

SimLex-999 (Hill et al., 2015)

Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)

TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

# Evaluating embeddings

Compare to human scores on word similarity-type tasks:

WordSim-353 (Finkelstein et al., 2002)

SimLex-999 (Hill et al., 2015)

Stanford Contextual Word Similarity (SCWS) dataset (Huang et al., 2012)

TOEFL dataset: *Levied is closest in meaning to: imposed, believed, requested, correlated*

# Properties of embeddings

Similarity depends on window size C

C =  $\pm 2$  The nearest words to *Hogwarts*:

*Sunnydale*

*Evernight*

C =  $\pm 5$  The nearest words to *Hogwarts*:

*Dumbledore*

*Malfoy*

*halfblood*

# Intrinsic Evaluation

- Relatedness:  
correlation  
(Spearman/Pearson)  
between vector  
similarity of pair of  
words and human  
judgments

word 1	word 2	human score
midday	noon	9.29
journey	voyage	9.29
car	automobile	8.94
...	...	...
professor	cucumber	0.31
king	cabbage	0.23

WordSim-353 (Finkelstein et al. 2002)

# Intrinsic Evaluation

- Analogical reasoning (Mikolov et al. 2013). For analogy  $\text{Germany} : \text{Berlin} :: \text{France} : ???$ , find closest vector to  $v(\text{"Berlin"}) - v(\text{"Germany"}) + v(\text{"France"})$

			target
possibly	impossibly	certain	uncertain
generating	generated	shrinking	shrank
think	thinking	look	looking
Baltimore	Maryland	Oakland	California
shrinking	shrank	slowing	slowed
Rabat	Morocco	Astana	Kazakhstan

# Analogical inference

- Mikolov et al. 2013 show that vector representations have some potential for analogical reasoning through vector arithmetic.

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

# Analogical inference

- Mikolov et al. 2013 show that vector representations have some potential for analogical reasoning through vector arithmetic.

king - man + woman  $\approx$  queen

a:b :: c:?

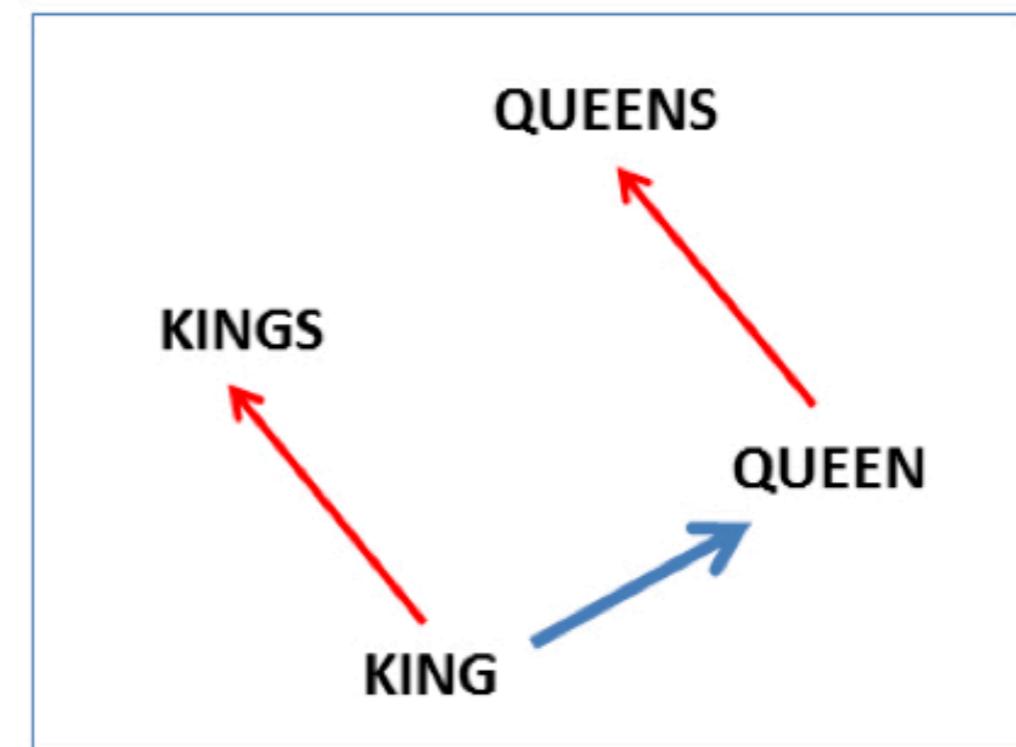
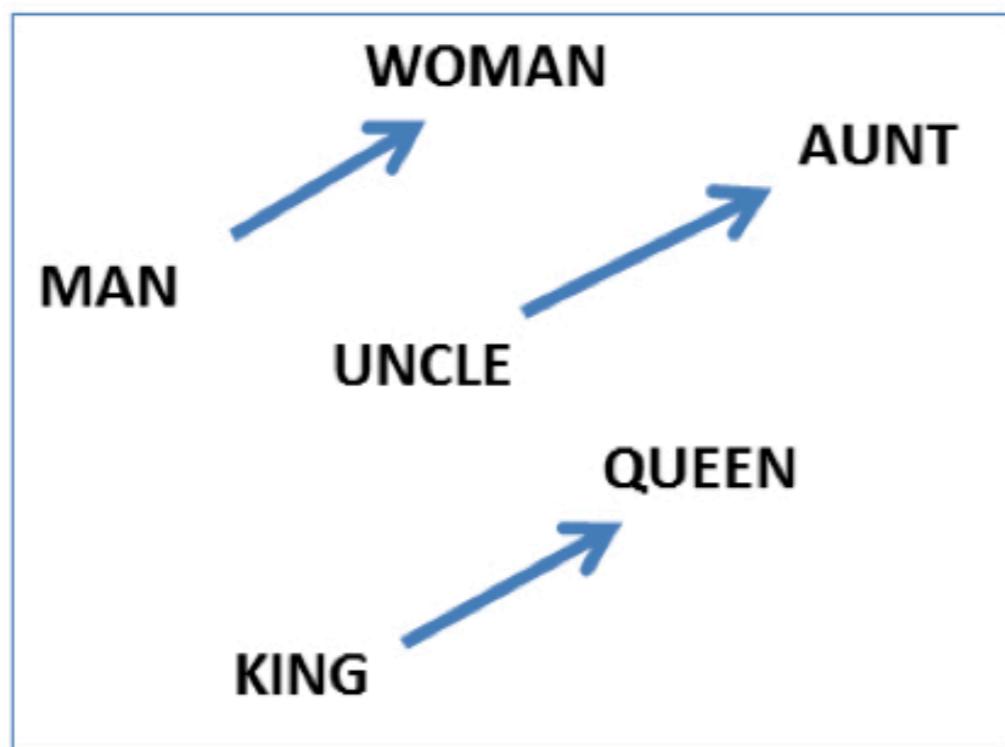


$$d = \arg \max_x \frac{(w_b - w_a + w_c)^T w_x}{\|w_b - w_a + w_c\|}$$

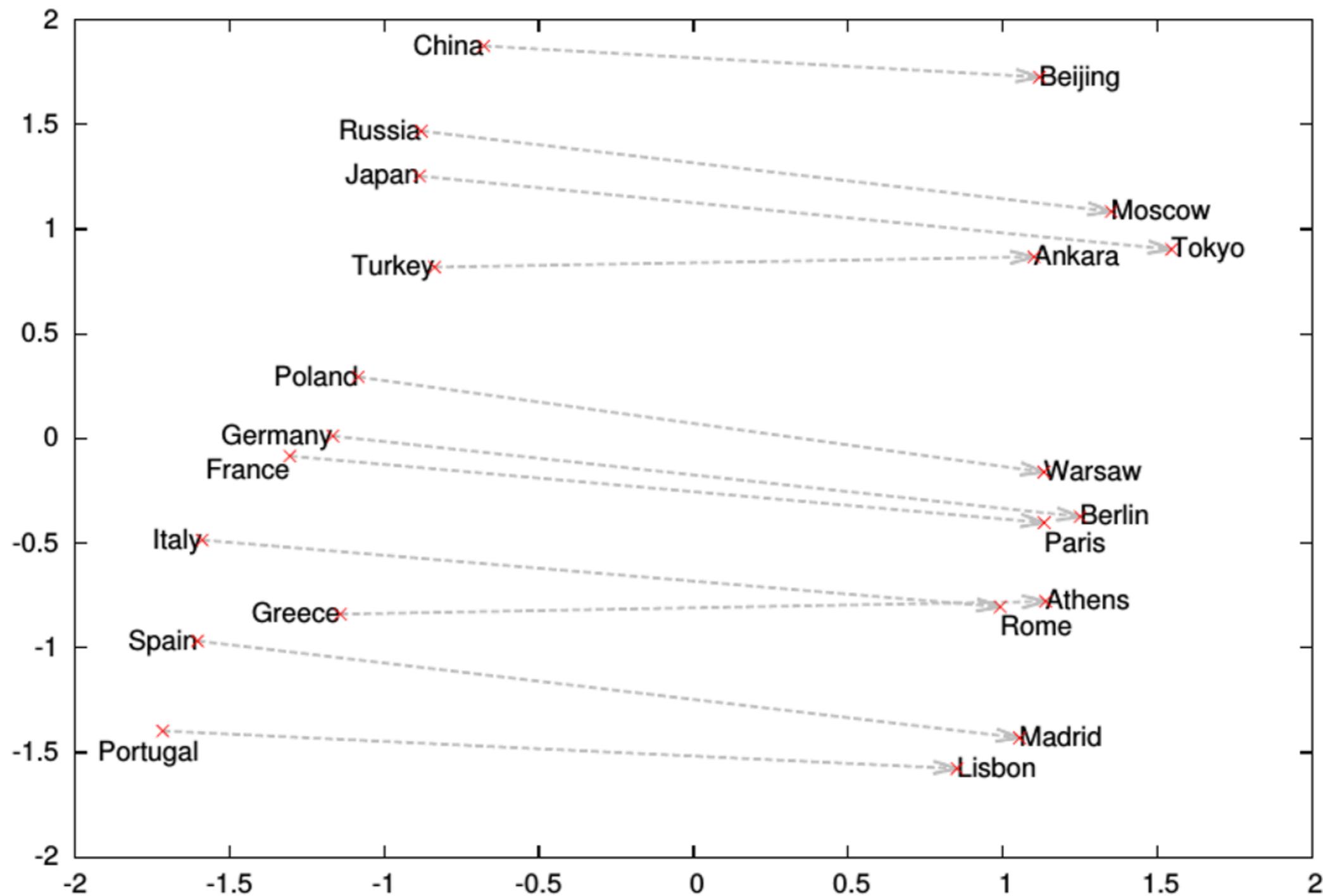
# Analogy: Embeddings capture relational meaning!

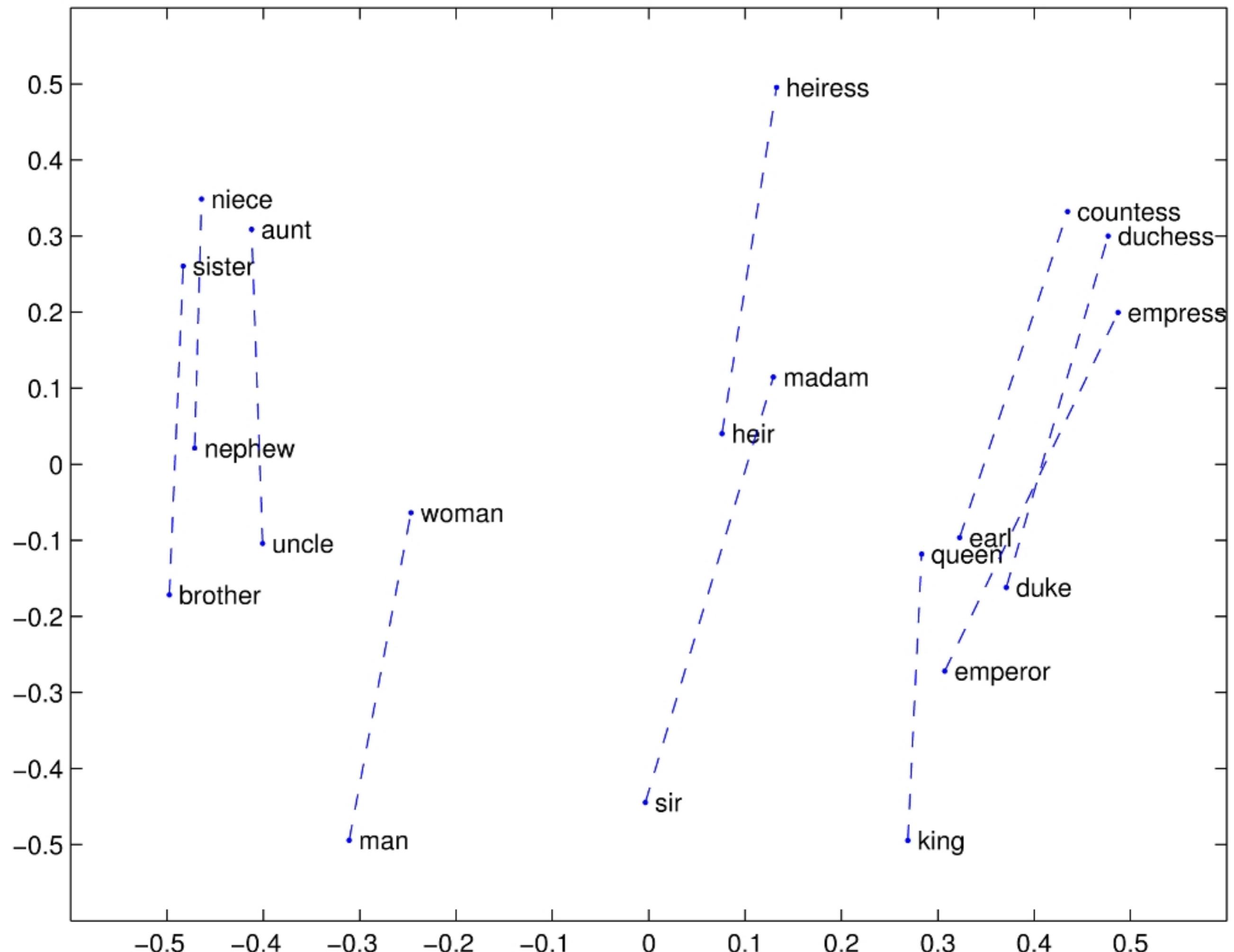
$\text{vector('king')} - \text{vector('man')} + \text{vector('woman')} = \text{vector('queen')}$

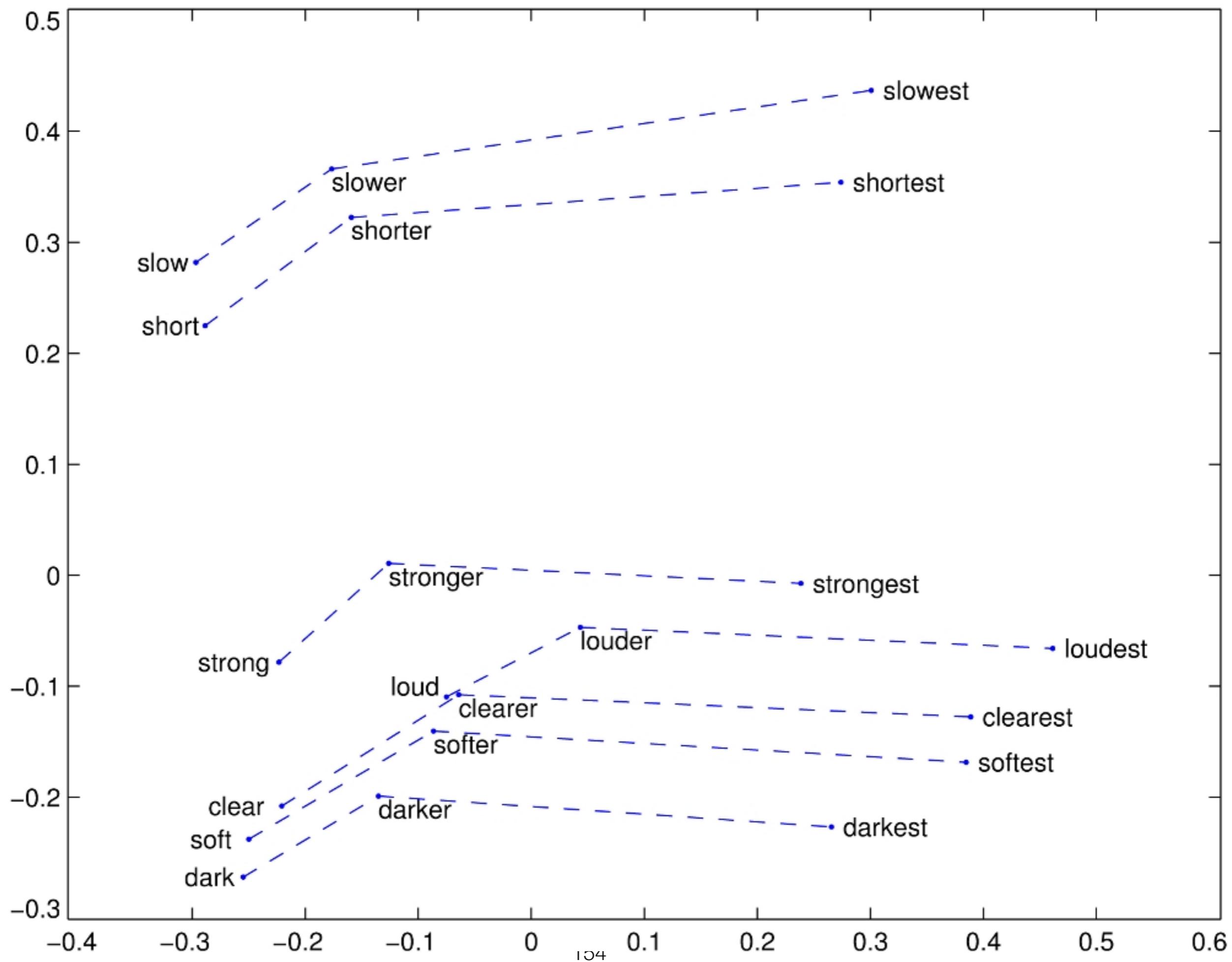
$\text{vector('Paris')} - \text{vector('France')} + \text{vector('Italy')} = \text{vector('Rome')}$



# Word analogies







We can use this regularity to  
learn regular morphology!

We can use this regularity to learn regular morphology!

apple - apples ≈ car - cars

We can use this regularity to learn regular morphology!

apple - apples  $\approx$  car - cars

invite - invitation  $\approx$  captivate - captivation

We can use this regularity to learn regular morphology!

apple - apples  $\approx$  car - cars

invite - invitation  $\approx$  captivate - captivation

run - running  $\approx$  swim - swimming

We can use this regularity to learn regular morphology!

apple - apples  $\approx$  car - cars

invite - invitation  $\approx$  captivate - captivation

run - running  $\approx$  swim - swimming

**We can even infer vectors!**

We can use this regularity to learn regular morphology!

apple - apples  $\approx$  car - cars

invite - invitation  $\approx$  captivate - captivation

run - running  $\approx$  swim - swimming

**We can even infer vectors!**

orange - oranges + pomegranate  $\approx$

# We can use this regularity to learn regular morphology!

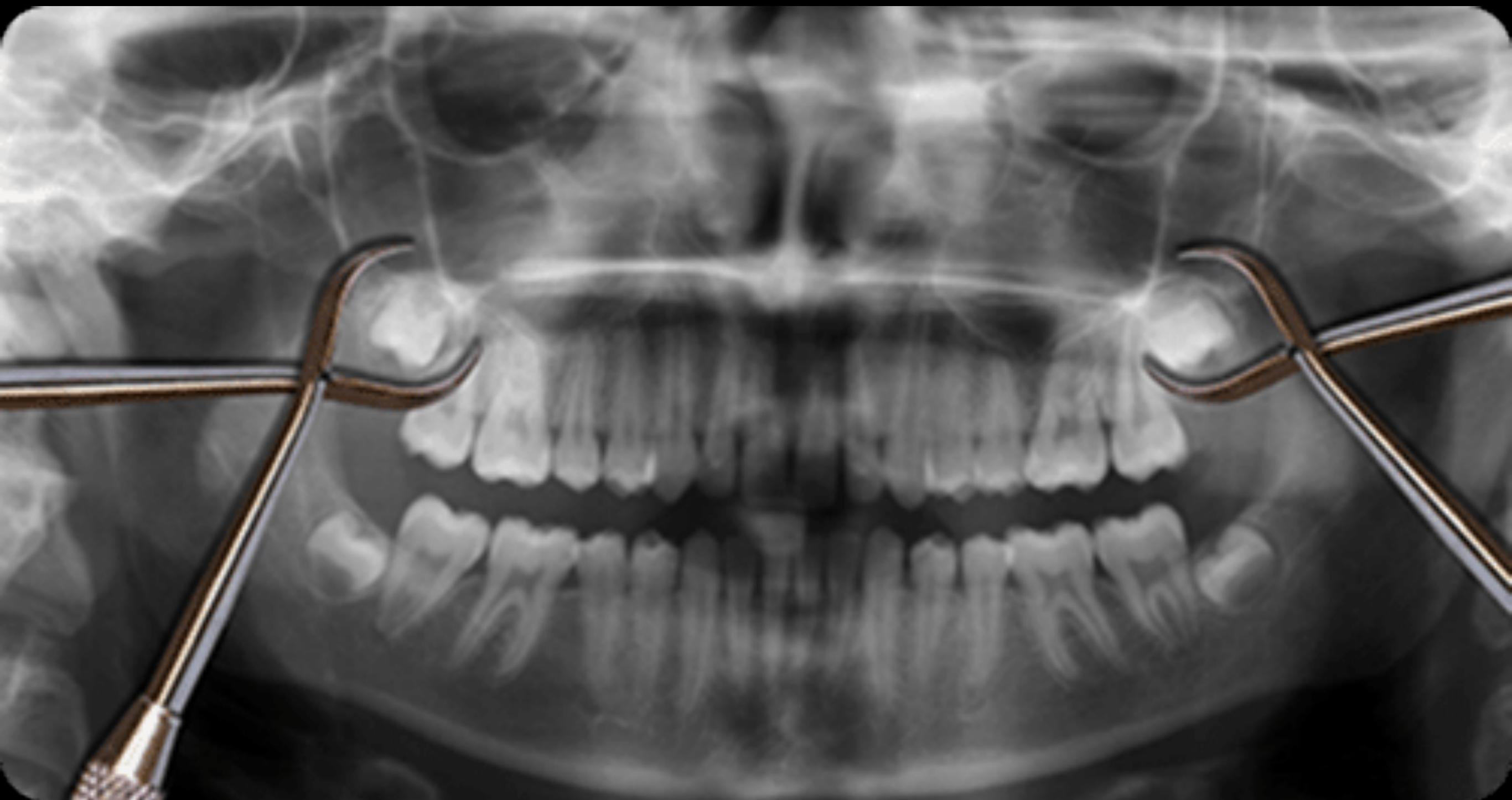
apple - apples  $\approx$  car - cars

invite - invitation  $\approx$  captivate - captivation

run - running  $\approx$  swim - swimming

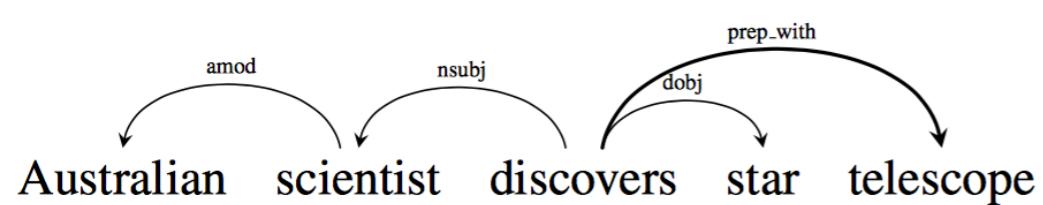
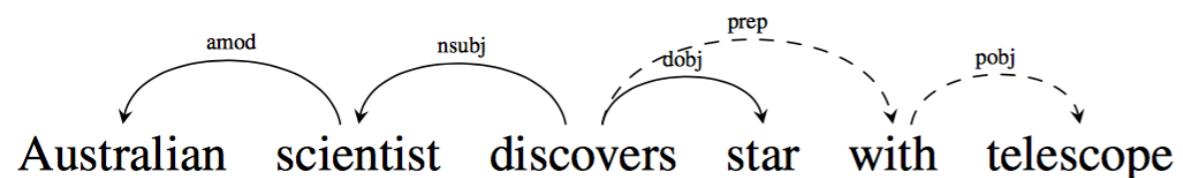
**We can even infer vectors!**

orange - oranges + pomegranate  $\approx$  pomegranates



# **Applications and Extensions**

# Use syntactic context



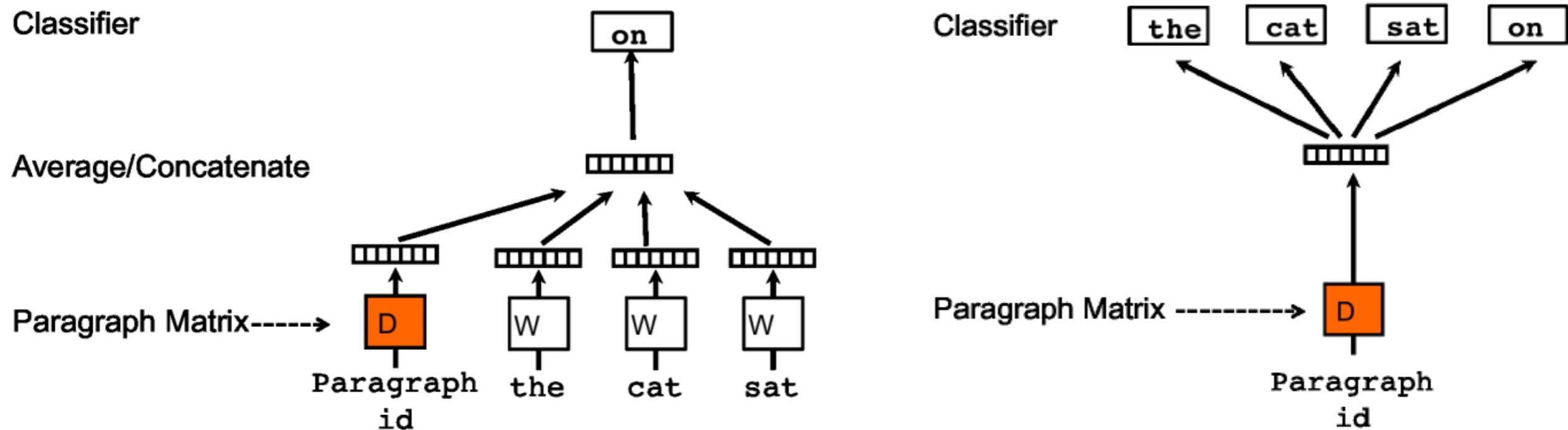
WORD	CONTEXTS
australian	scientist/amod <sup>-1</sup>
scientist	australian/amod, discovers/nsubj <sup>-1</sup>
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj <sup>-1</sup>
telescope	discovers/prep_with <sup>-1</sup>

Lin 1998; Levy and Goldberg 2014

Target Word	BoW5	BoW2	DEPS
batman	nightwing aquaman catwoman superman manhunter	superman superboy catwoman batgirl	superman superboy supergirl catwoman aquaman
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood	sunnydale collinwood calarts greendale millfield
turing	nondeterministic non-deterministic computability deterministic finite-state	non-deterministic finite-state nondeterministic buchi primality	paulling hotelling heting lessing hamming
florida	gainesville fla jacksonville tampa lauderdale	fla alabama gainesville tallahassee texas	texas louisiana georgia california carolina
object-oriented	aspect-oriented smalltalk event-driven prolog domain-specific	aspect-oriented event-driven objective-c dataflow 4gl	event-driven domain-specific rule-based data-driven human-centered
dancing	singing dance dances dancers tap-dancing	singing dance dances breakdancing clowning	singing rapping breakdancing miming busking

# Why stop at words? Embed the meaning of sentence/text!

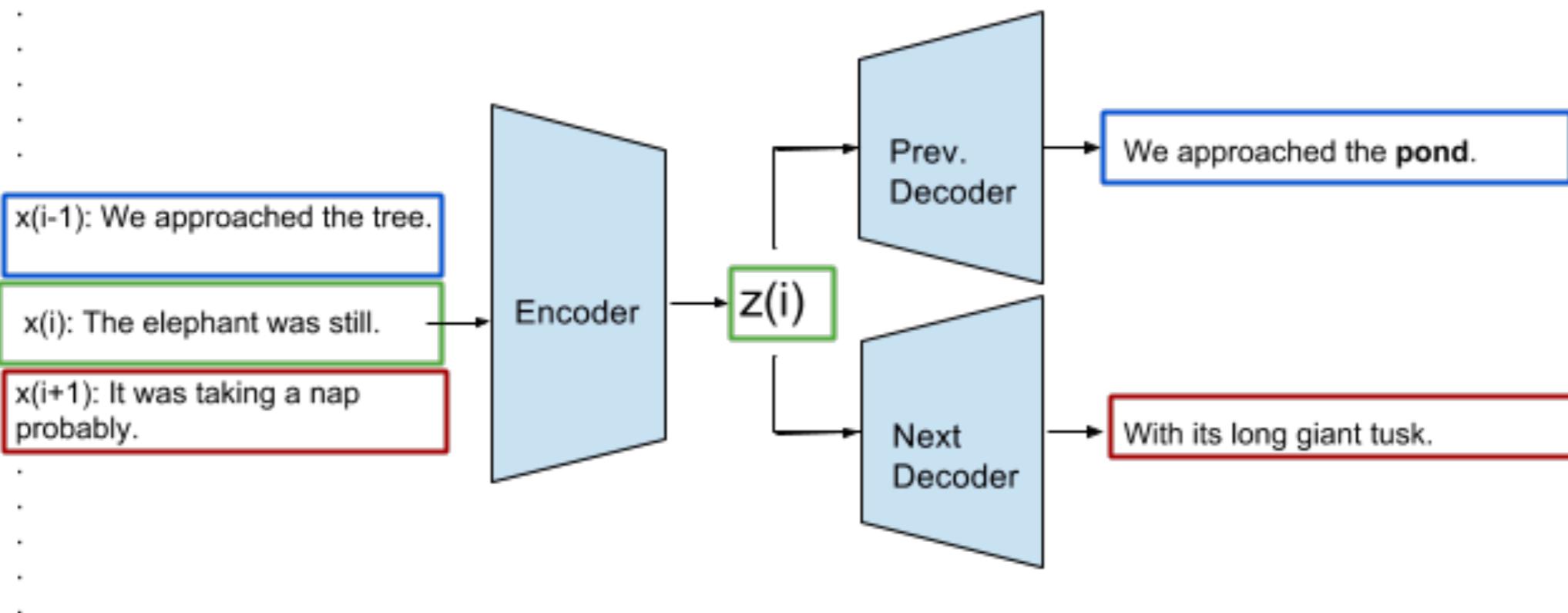
- Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input



# Skip-thought vectors (cf. Skip-gram)

x(0): Hi, My name is Sanyam

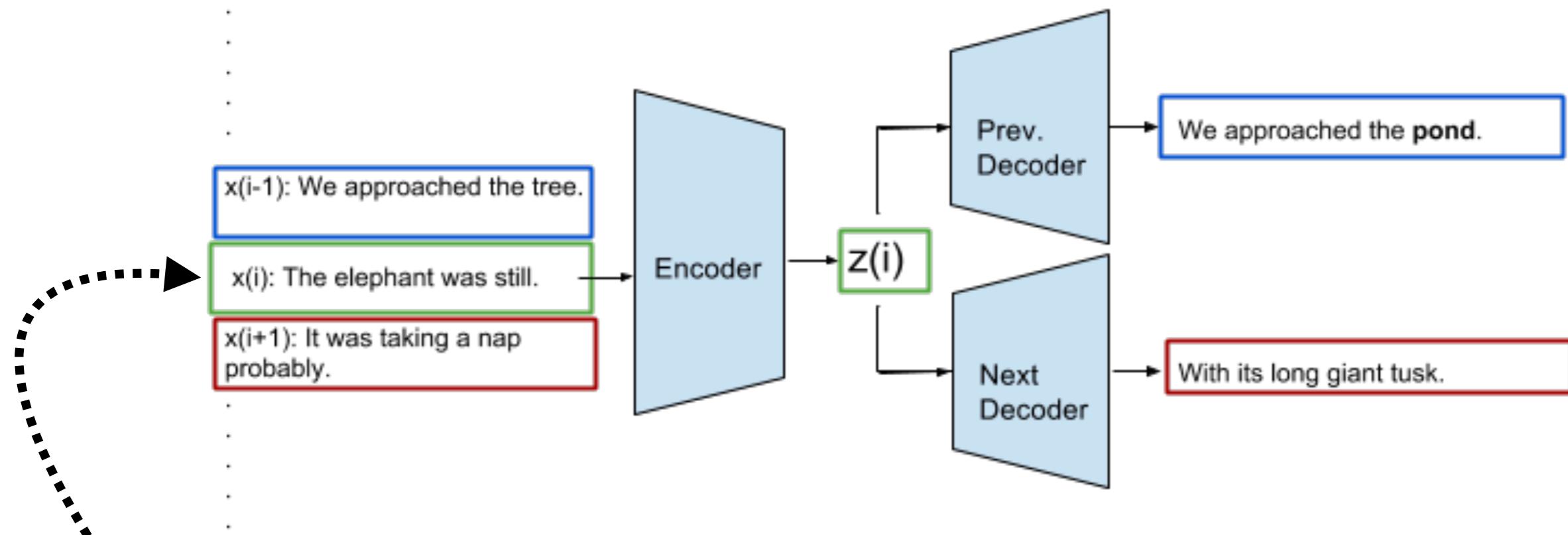
x(1): Today, I went to the zoo.



# Skip-thought vectors (cf. Skip-gram)

x(0): Hi, My name is Sanyam

x(1): Today, I went to the zoo.

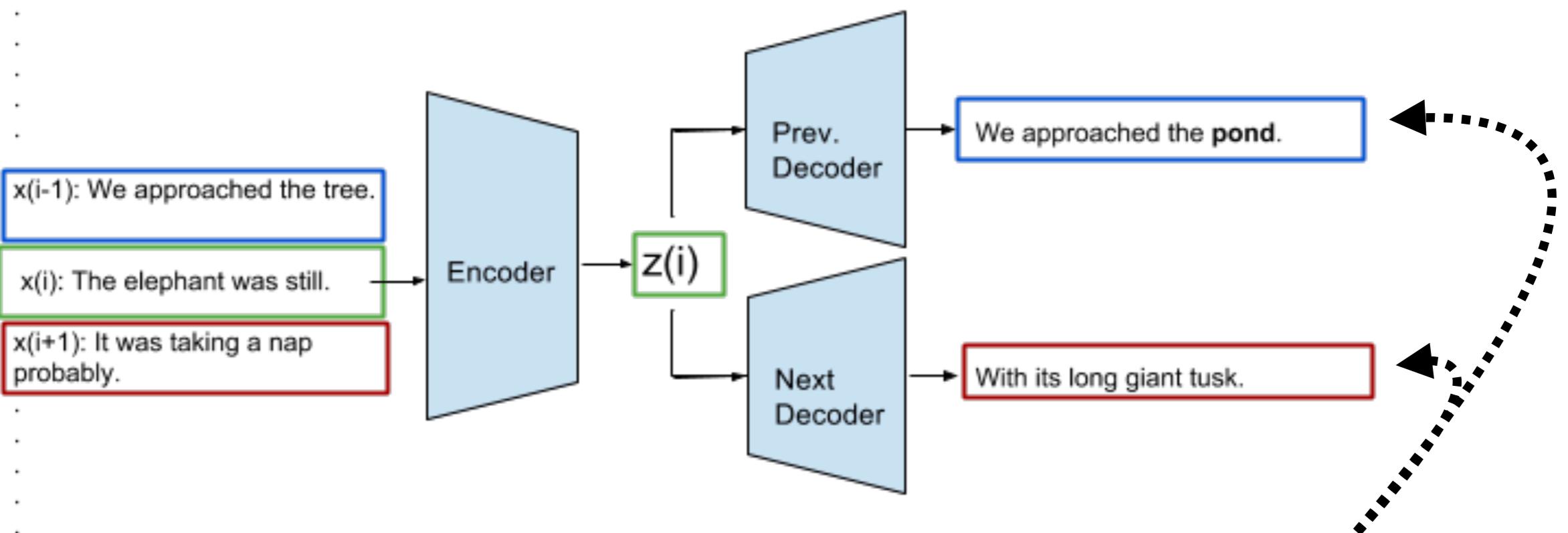


Embed a whole sentence

# Skip-thought vectors (cf. Skip-gram)

x(0): Hi, My name is Sanyam

x(1): Today, I went to the zoo.



Embed a **whole sentence**

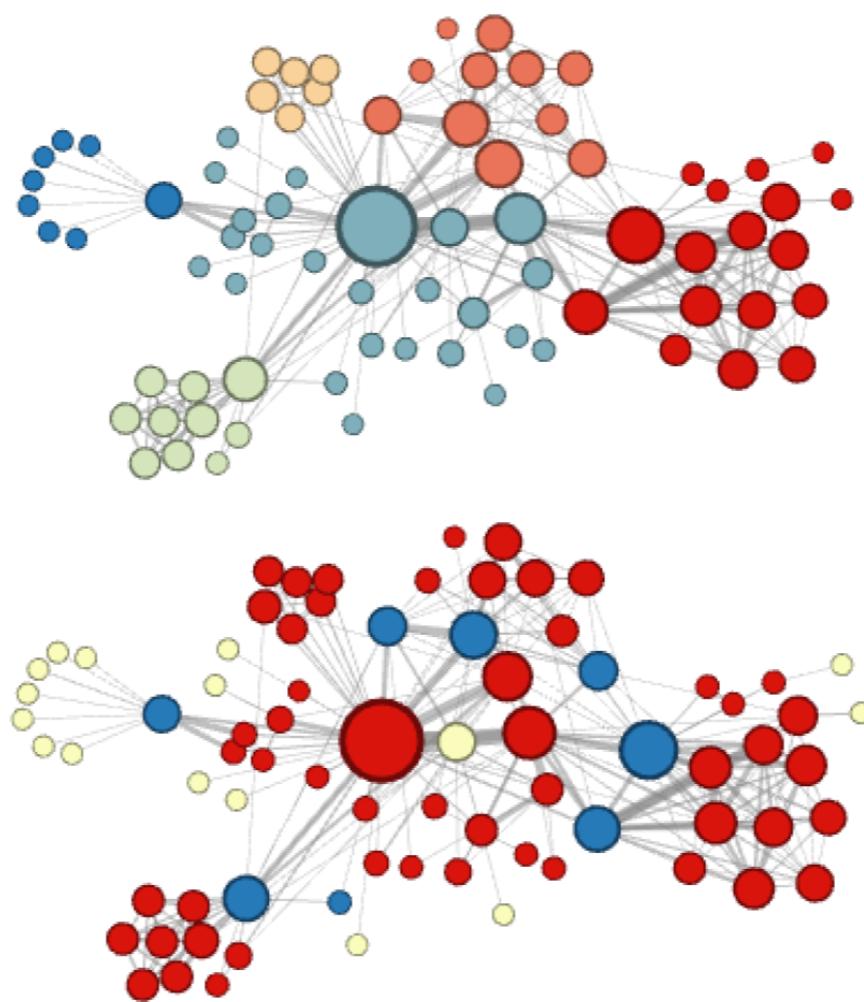
Predict the sentence embeddings for the **preceding** and **following** sentences

# emoji2vec



Eisner et al. (2016), “emoji2vec: Learning Emoji Representations from their Description”

# node2vec



Grover and Leskovec (2016), “node2vec: Scalable Feature Learning for Networks”

# Trained embeddings

- Word2vec  
<https://code.google.com/archive/p/word2vec/>
- Glove  
<http://nlp.stanford.edu/projects/glove/>
- Levy/Goldberg dependency embeddings  
<https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings/>

# Low-dimensional distributed representations

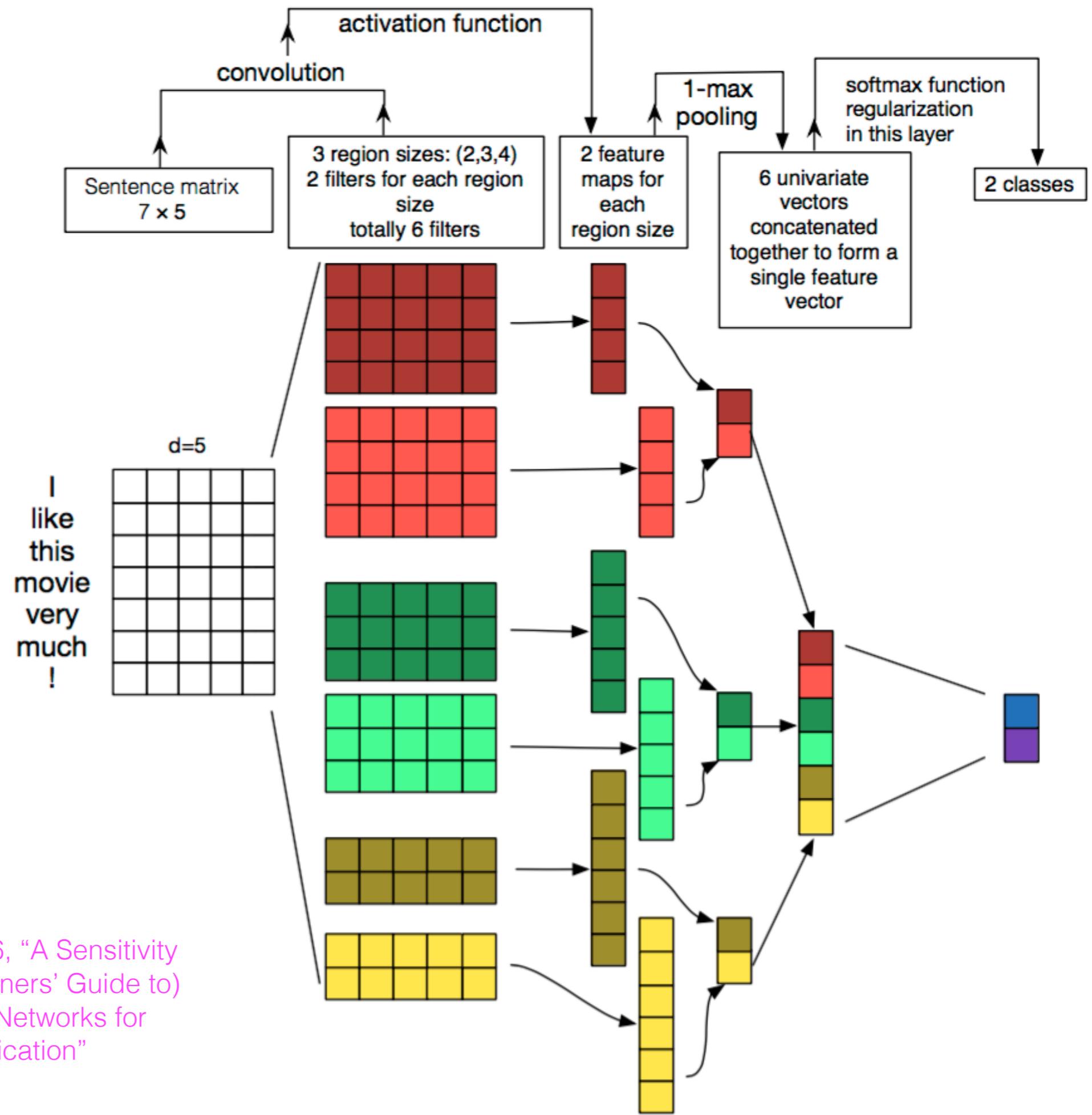
- Low-dimensional, dense word representations are extraordinarily powerful (and are arguably responsible for much of gains that neural network models have in NLP).
- Lets your representation of the input share statistical strength with words that behave similarly in terms of their distributional properties (often **synonyms** or words that belong to the same **class**).

# Two kinds of “training” data

- The labeled data for a specific task (e.g., labeled sentiment for movie reviews): ~ 2K labels/reviews, ~1.5M words → used to train a supervised model
- General text (Wikipedia, the web, books, etc.), ~ trillions of words → used to train word distributed representations

# Using dense vectors

- In neural models (CNNs, RNNs, LM), replace the  $V$ -dimensional sparse vector with the much smaller  $K$ -dimensional dense one.
- Can also take the derivative of the loss function with respect to those representations to optimize for a particular task.



Zhang and Wallace 2016, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification”

# Using dense vectors

- (Short) document-level representation: coordinate-wise max, min or average; use directly in neural network [Joulin et al. 2016]
- K-means clustering on vectors into distinct partitions (though beware of strange geometry [Mimno and Thompson 2017])

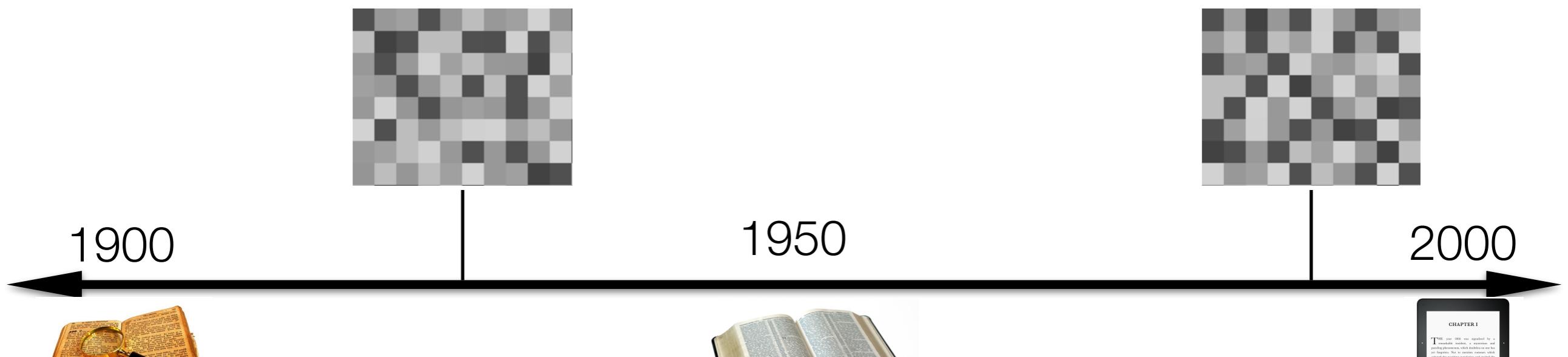
# Embeddings can help study word history!

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

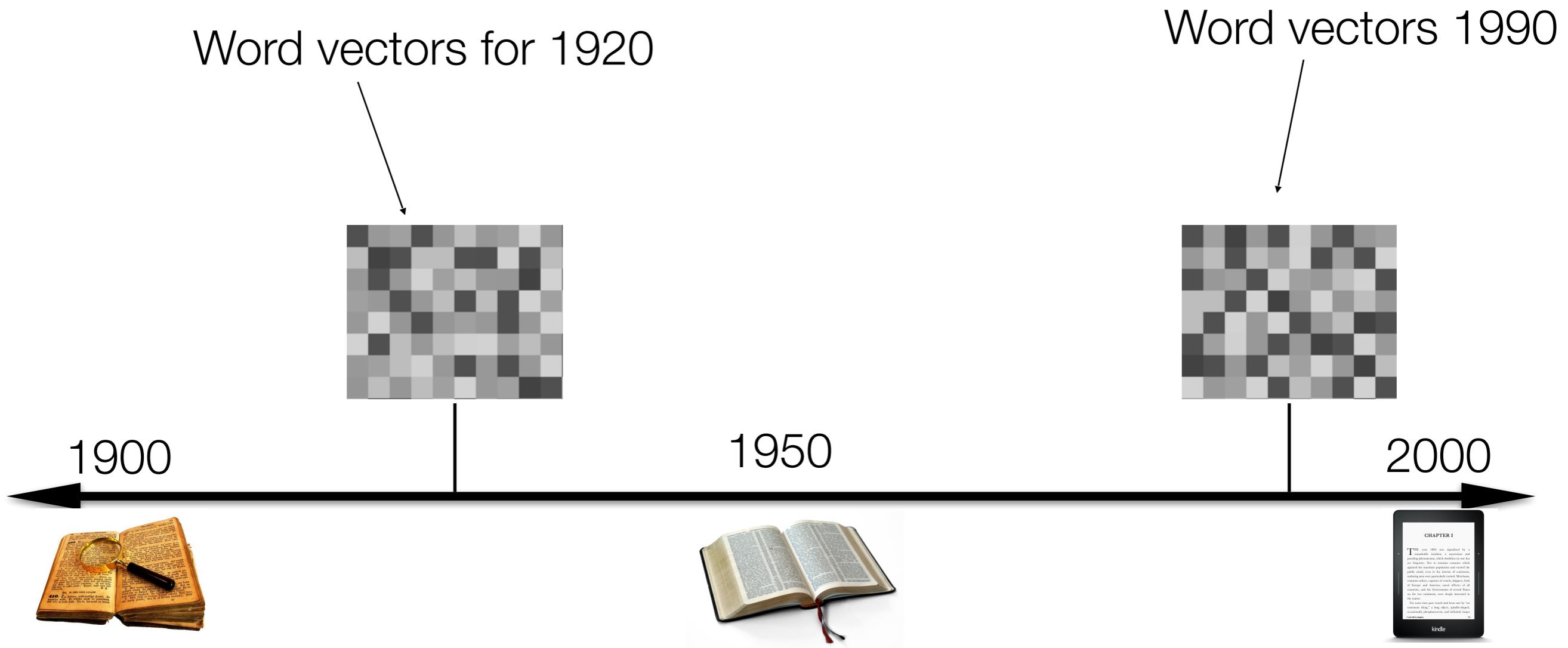


Will Hamilton

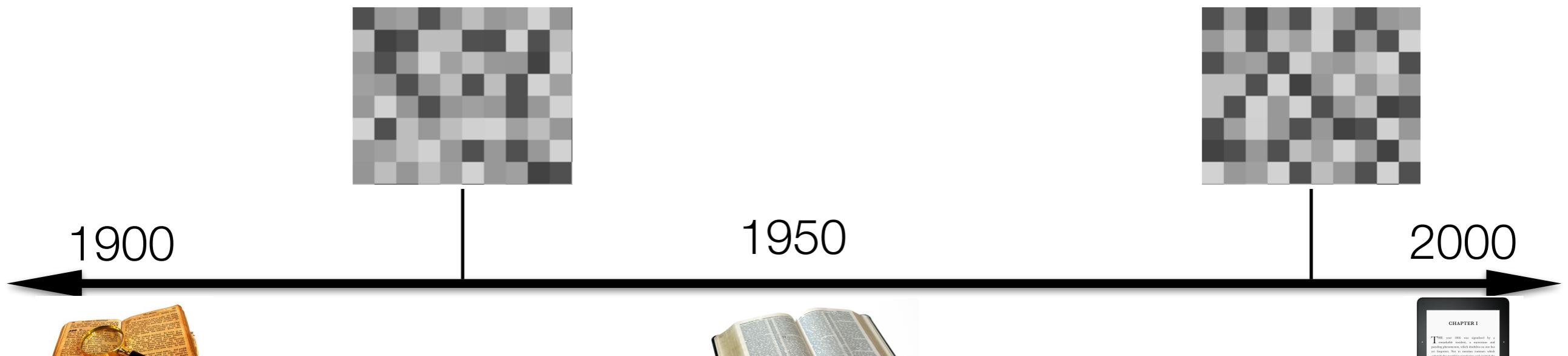
# Diachronic word embeddings for studying language change!



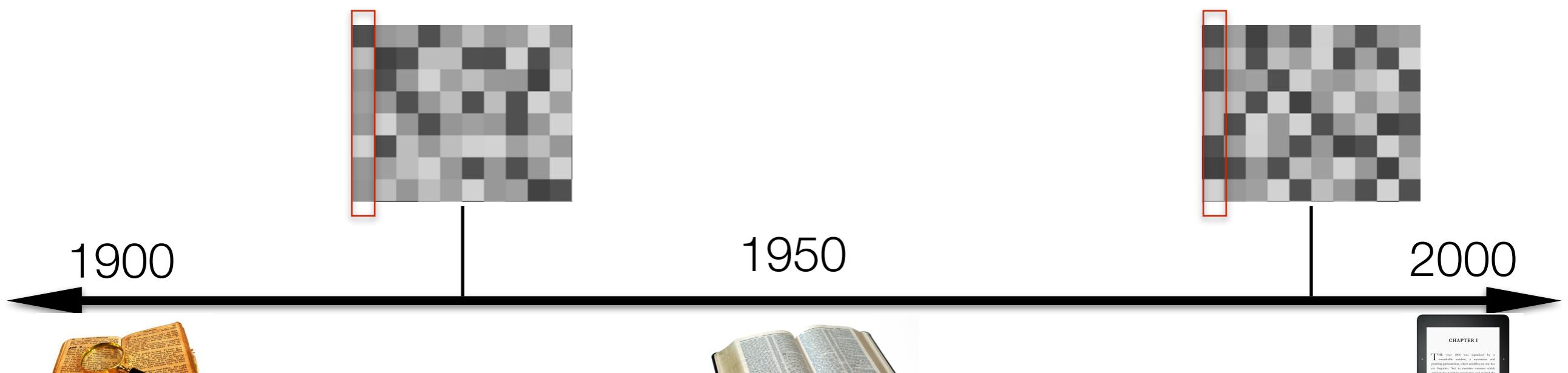
# Diachronic word embeddings for studying language change!



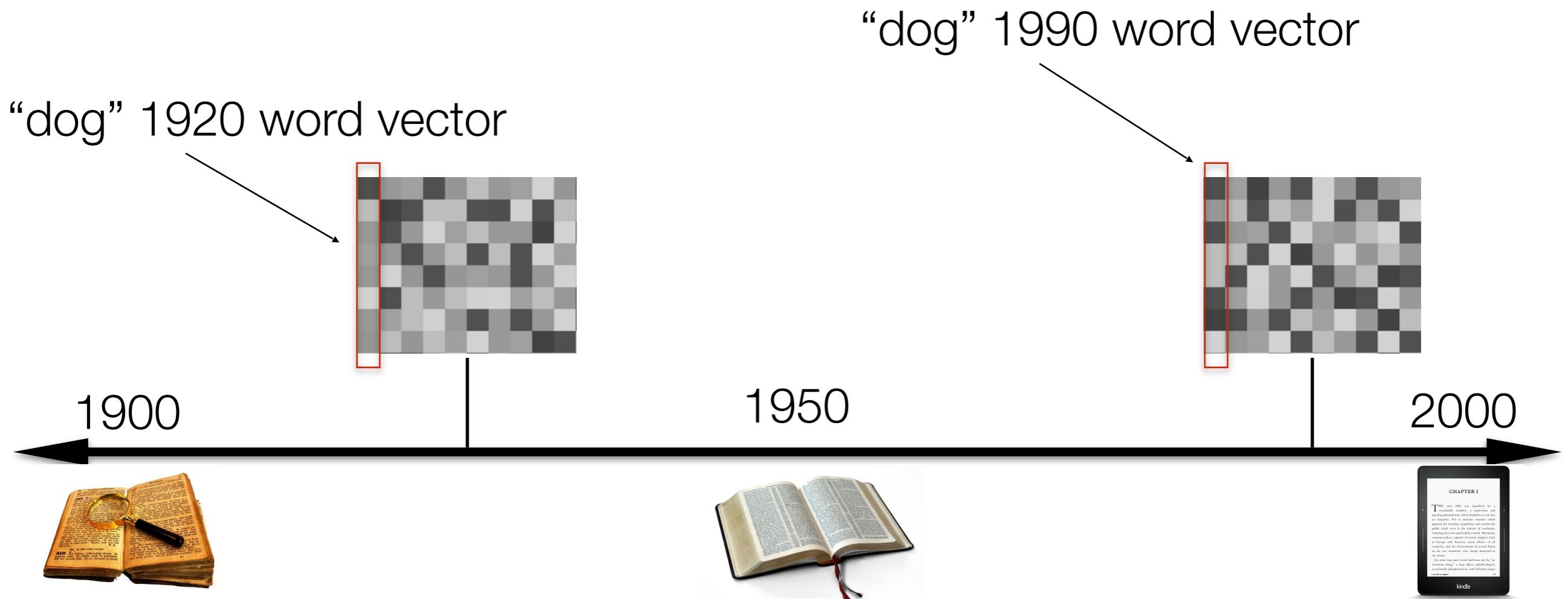
# Diachronic word embeddings for studying language change!



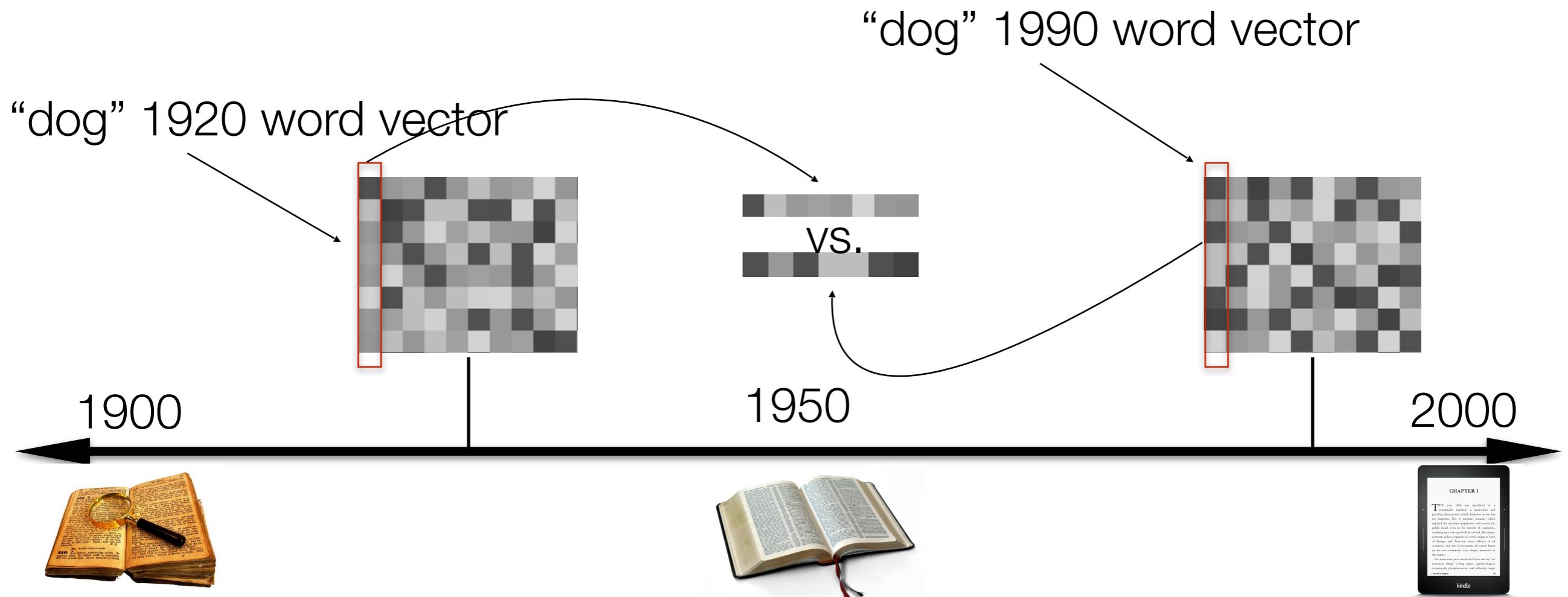
# Diachronic word embeddings for studying language change!



# Diachronic word embeddings for studying language change!

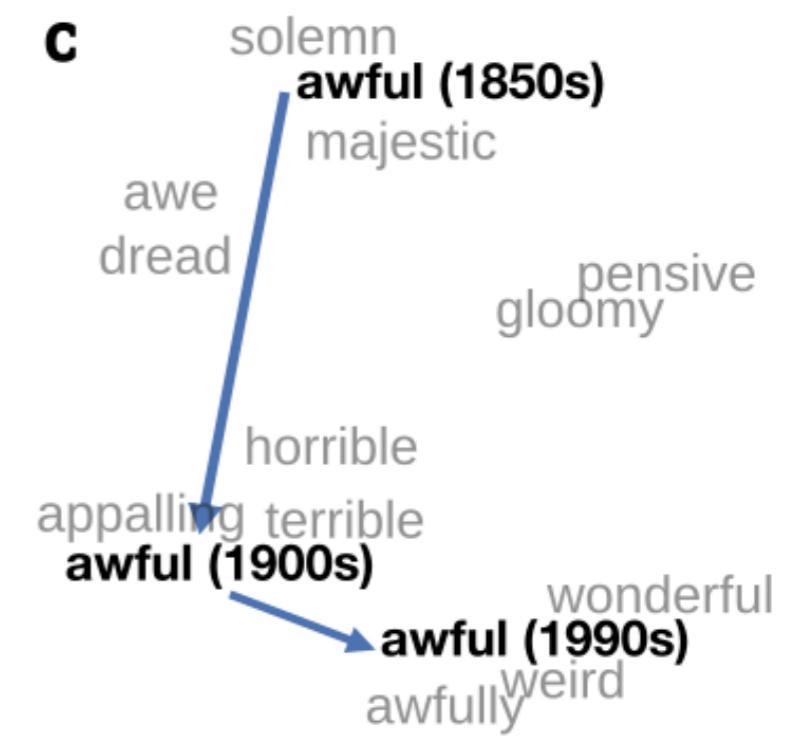
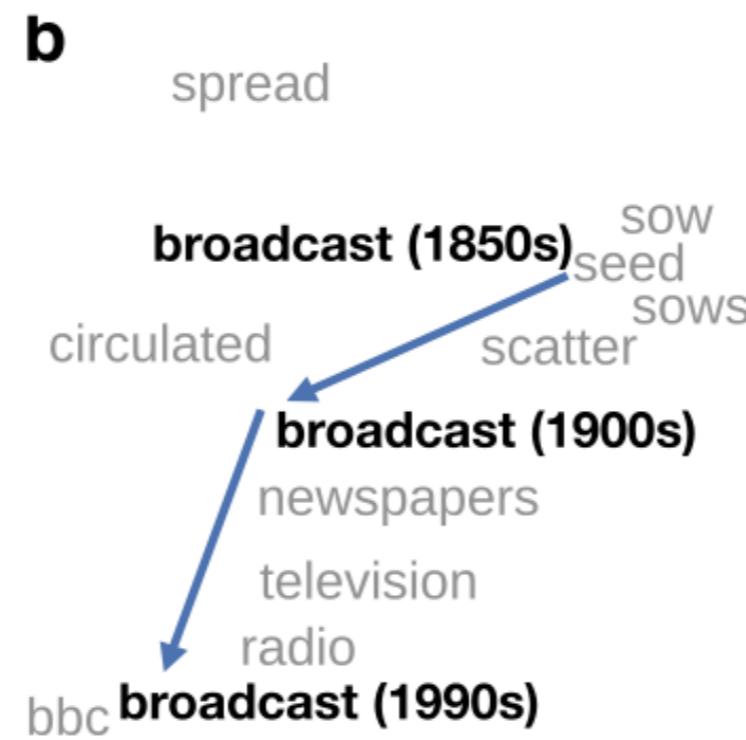
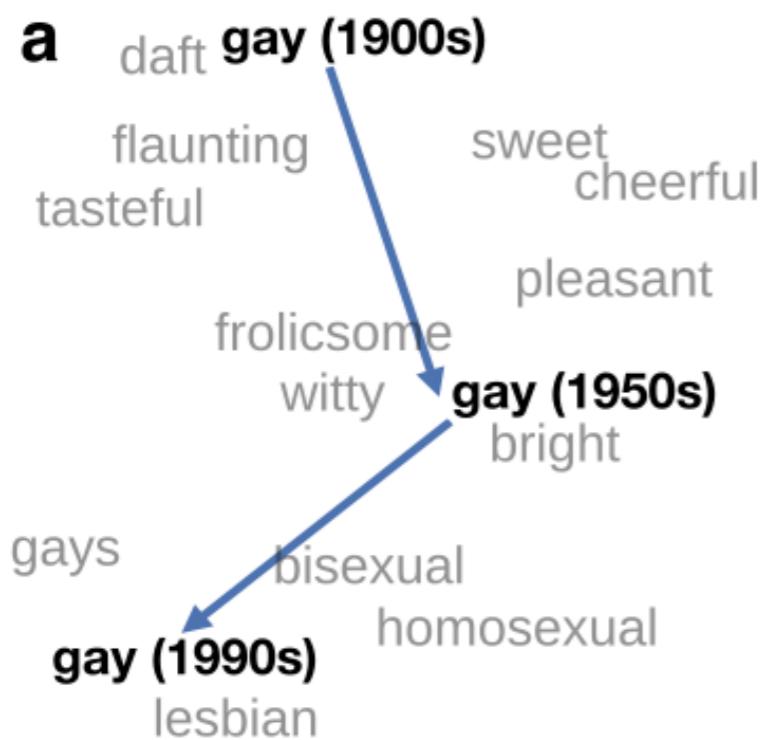


# Diachronic word embeddings for studying language change!



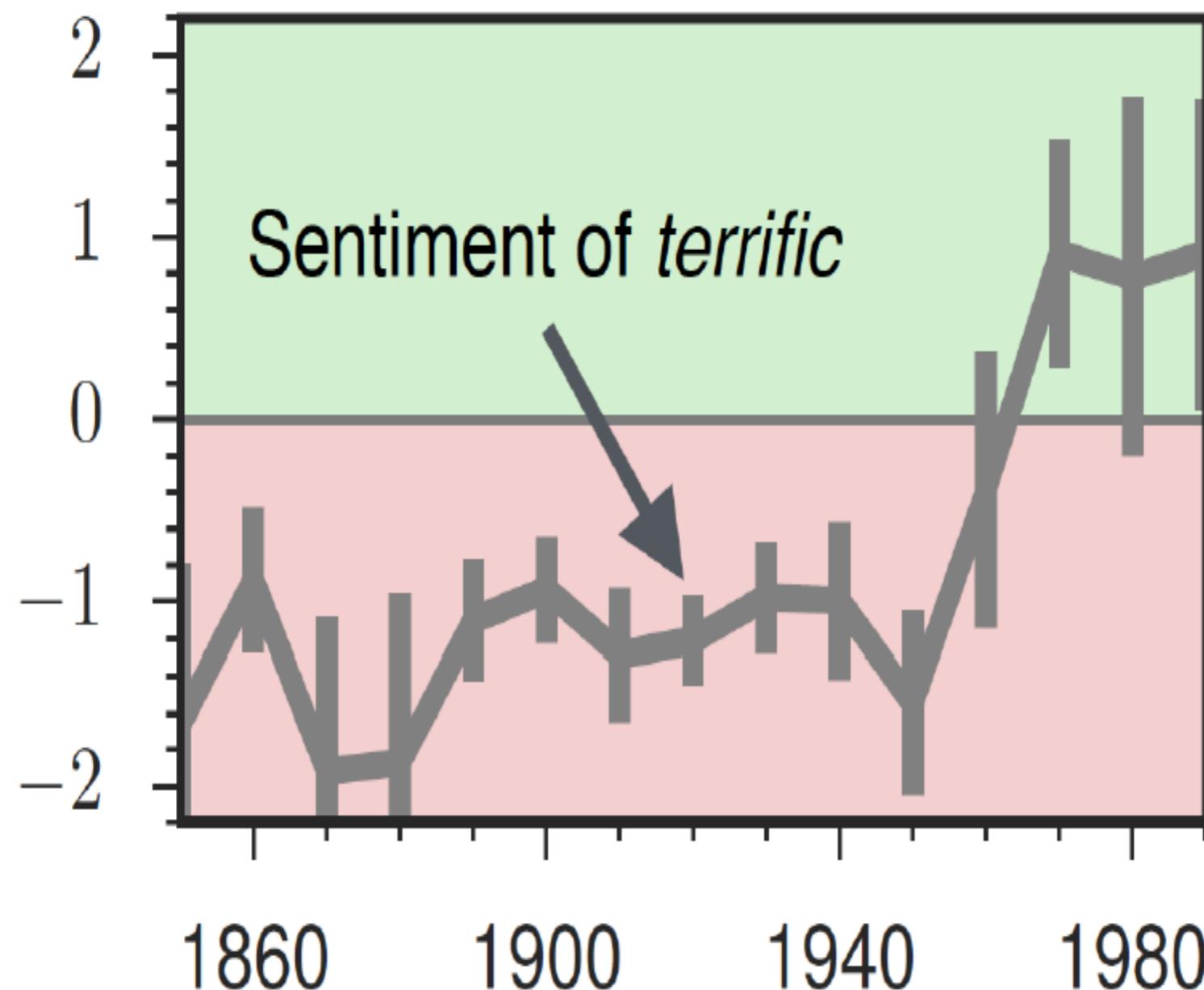
# Visualizing changes

Project 300 dimensions down into 2



# The evolution of sentiment words

Negative words change faster than positive words



# Science

AAAS

Home

News

Journals

Topics

Careers

Science

Science Advances

Science Immunology

Science Robotics

Science Signaling

Science Translational Medicine

SHARE

REPORT



0



Aylin Caliskan<sup>1,\*</sup>, Joanna J. Bryson<sup>1,2,\*</sup>, Arvind Narayanan<sup>1,\*</sup>

<sup>+</sup> See all authors and affiliations



13

Science 14 Apr 2017:  
Vol. 356, Issue 6334, pp. 183-186  
DOI: 10.1126/science.aal4230



Peer Reviewed  
← see details

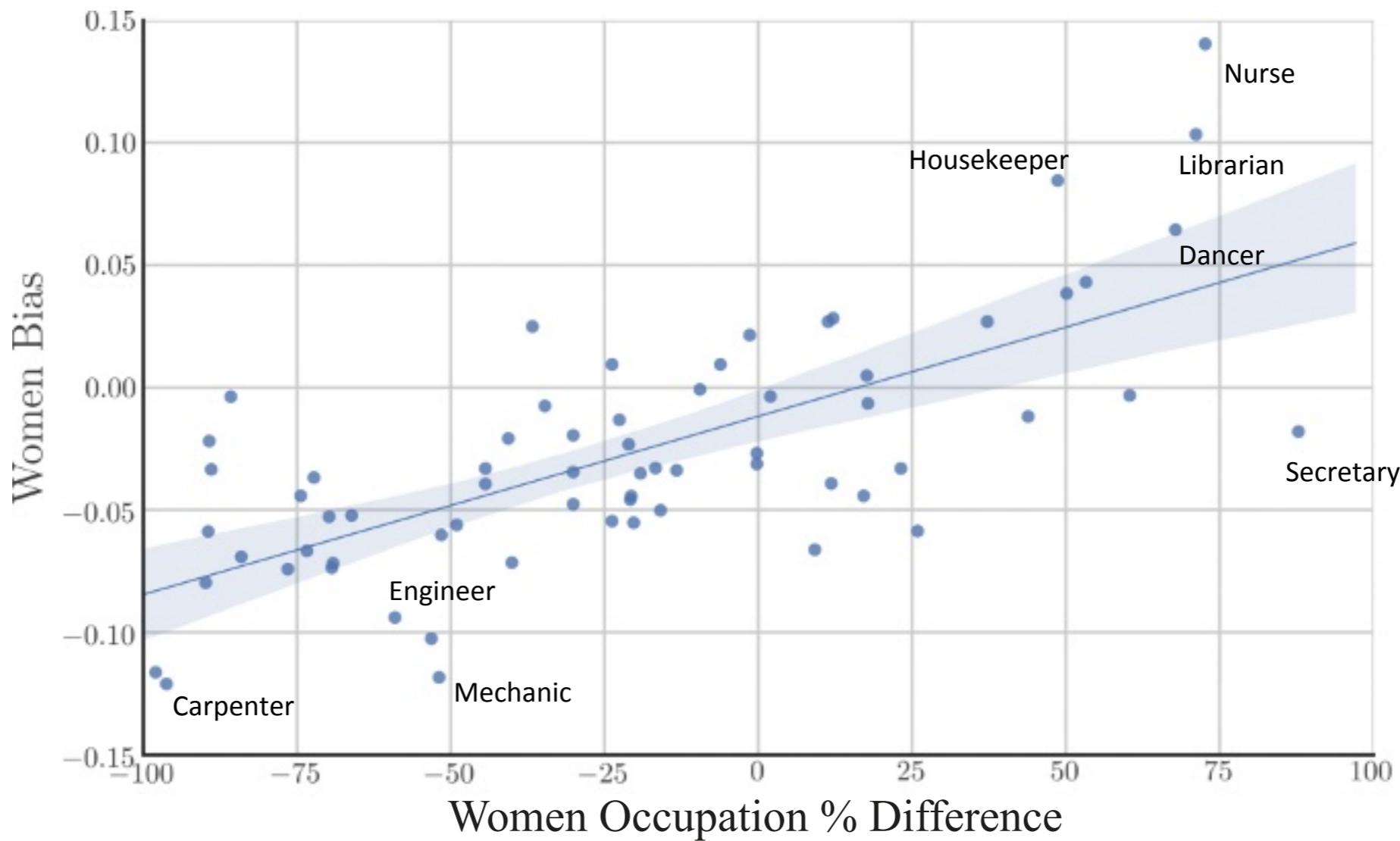
Article

Figures & Data

Info & Metrics

eLetters

PDF



**Fig. 1.** Women's occupation relative percentage vs. embedding bias in Google News vectors. More positive indicates more associated with women on both axes.  $P < 10^{-10}$ ,  $r^2 = 0.499$ . The shaded region is the 95% bootstrapped confidence interval of the regression line. In this single embedding, then, the association in the embedding effectively captures the percentage of women in an occupation.



# Summary of what we learned today

# Summary of what we learned today

- We can learn “word meaning” by observing what kinds of contexts a word appears in

# Summary of what we learned today

- We can learn “word meaning” by observing what kinds of contexts a word appears in
- Dimensionality reduction can improve representations by making similar representations more similar and by reducing space

# Summary of what we learned today

- We can learn “word meaning” by observing what kinds of contexts a word appears in
- Dimensionality reduction can improve representations by making similar representations more similar and by reducing space
- Learning to predict context words (deep learning) performs better than counting context words

# Open Research Questions

# Open Research Questions

- How to construct vectors for unknown words, especially using characters or morphemes

# Open Research Questions

- How to construct vectors for unknown words, especially using characters or morphemes
- How to learn word vectors in different language that are comparable “cats” (EN)  $\approx$  “gatos” (ES)

# Open Research Questions

- How to construct vectors for unknown words, especially using characters or morphemes
- How to learn word vectors in different language that are comparable “cats” (EN)  $\approx$  “gatos” (ES)
- How to tune vectors for specific tasks and how to make general purpose vectors

# Open Research Questions

- How to construct vectors for unknown words, especially using characters or morphemes
- How to learn word vectors in different language that are comparable “cats” (EN)  $\approx$  “gatos” (ES)
- How to tune vectors for specific tasks and how to make general purpose vectors
- New architectures or techniques for creating word vectors that perform better on downstream tasks

# What you should know

# What you should know

- How to construct a basic word vector

# What you should know

- How to construct a basic word vector
- What is the TF-IDF and how to calculate it

# What you should know

- How to construct a basic word vector
- What is the TF-IDF and how to calculate it
- The basics of SVD and why/when we should use it

# What you should know

- How to construct a basic word vector
- What is the TF-IDF and how to calculate it
- The basics of SVD and why/when we should use it
- How word2vec works



# Homework 2: You get to implement word2vec!

# Homework 2: You get to implement word2vec!

- We're giving you skeleton code for the basic parts and you get to fill in the learning part

# Homework 2: You get to implement word2vec!

- We're giving you skeleton code for the basic parts and you get to fill in the learning part
- You'll do deep learning 😊 😊 😊 😊 😊 😊

# Homework 2: You get to implement word2vec!

- We're giving you skeleton code for the basic parts and you get to fill in the learning part
- You'll do **deep learning** 😊 😊 😊 😊 😊 😊
- You'll get to evaluate your learned vectors on simple word similarity tasks

# Project Proposals (1)

# Project Proposals (1)

- Three key parts:

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns
  - Good: Part of speech tag medical text

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns
  - Good: Part of speech tag medical text
  - Good: Generate convincing endings to stories

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns
  - Good: Part of speech tag medical text
  - Good: Generate convincing endings to stories
  - Bad: Look for differences in documents

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns
  - Good: Part of speech tag medical text
  - Good: Generate convincing endings to stories
  - Bad: Look for differences in documents
  - Bad: Run topic modeling on data

# Project Proposals (1)

- Three key parts:
- 1: A **measurable research\* Question**: What are you tryin to do
  - Good: Classify online message as state-sponsored campaigns
  - Good: Part of speech tag medical text
  - Good: Generate convincing endings to stories
  - Bad: Look for differences in documents
  - Bad: Run topic modeling on data
  - Bad: Run a sentiment analysis system on some data

# Project Proposals (2)

# Project Proposals (2)

- Three key parts:

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - Good: We'll use F1 to score ...

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - Good: We'll use F1 to score ...
  - Good: The parser will be evaluated using UAS

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - Good: We'll use F1 to score ...
  - Good: The parser will be evaluated using UAS
  - Good: We'll compare human judgments (generated by us) on 1000 examples with machine predictions

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - **Good**: We'll use F1 to score ...
  - **Good**: The parser will be evaluated using UAS
  - **Good**: We'll compare human judgments (generated by us) on 1000 examples with machine predictions
  - **Bad**: I'll look the clusters

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - **Good**: We'll use F1 to score ...
  - **Good**: The parser will be evaluated using UAS
  - **Good**: We'll compare human judgments (generated by us) on 1000 examples with machine predictions
  - **Bad**: I'll look the clusters
  - **Bad**: I'll crowdsource some data

# Project Proposals (2)

- Three key parts:
- 2: **An evaluation metric**: How will you know if you succeed
  - **Good**: We'll use F1 to score ...
  - **Good**: The parser will be evaluated using UAS
  - **Good**: We'll compare human judgments (generated by us) on 1000 examples with machine predictions
  - **Bad**: I'll look the clusters
  - **Bad**: I'll crowdsource some data
  - **Bad**: the model is already trained so I don't need to evaluate

# Project Proposals (3)

# Project Proposals (3)

- Three key parts:

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - Good: We use data set from Person et al. (2018)
  - Good: I already have this dataset from my work at XYZ

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.
  - **Bad**: I'm going to scrape some news websites

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.
  - **Bad**: I'm going to scrape some news websites
  - **Bad**: I will file a request to get data from this place and I'll wait to see

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.
  - **Bad**: I'm going to scrape some news websites
  - **Bad**: I will file a request to get data from this place and I'll wait to see
  - **Bad**: I found this data on Kaggle (etc.)

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.
  - **Bad**: I'm going to scrape some news websites
  - **Bad**: I will file a request to get data from this place and I'll wait to see
  - **Bad**: I found this data on Kaggle (etc.)
  - **Bad**: I will use this data from a paper that also used 8 GPUs for 6 months to...

# Project Proposals (3)

- Three key parts:
- 3: **A dataset**: What are you going to use?
  - **Good**: We use data set from Person et al. (2018)
  - **Good**: I already have this dataset from my work at XYZ
  - **Good**: Here is a very concrete plan on how I can get the data in two weeks.
  - **Bad**: I'm going to scrape some news websites
  - **Bad**: I will file a request to get data from this place and I'll wait to see
  - **Bad**: I found this data on Kaggle (etc.)
  - **Bad**: I will use this data from a paper that also used 8 GPUs for 6 months to...

**No Kaggle data or data with lots of public examples**

# Lots of places to find data

- <https://datasetsearch.research.google.com/>
- <https://msropendata.com/>
- Papers you'll read during class (we'll upload the whole list shortly)
- Ask us!

# Team Requirements

- Most people do single person teams
- Each person should add 1.25-persons worth of work  
—no easy teams
- Definitely not:
  - I'll train the Logistic Regression Classifier and they'll train the Naive Bayes
  - I'll extract the features and they'll train the classifiers

- We'll have additional office hours next week to talk about the project ideas
- feel free to post questions or requests for data on Piazza
- Up to two weeks with no penalty and gets rescored (up to once)