Lecture 6: Word Embeddings (contd.)

Instructor: Swabha Swayamdipta

USC CSCI 544 Applied NLP

Sep 12, Fall 2024

Some slides adapted from Dan Jurafsky and Chris Manning and Xuezhe Ma

Announcements + Logistics

Announcements + Logistics

- Quiz 1 grades released, regrade requests can be made within a week of grade release date
  - We will have opportunities later in class for you to make up your grade
- Group selection deadline
  - TAs have now been assigned to your groups - your TA advisors for your projects
  - 47 teams - check your team / TA assignments
  - Project Proposal Date is coming up (9/26)
- Next week:
  - HW1 due
  - HW2 release
  - Quiz 2

Lecture Outline

- Announcements
- Recap: Multinomial LR
- Recap: Lexical Semantics
- word2vec
- GloVe
- Properties and Evaluation of Word Embeddings

Recap: Multinomial LR

Multinomial Logistic Regression

The probability of everything must still sum to 1

$K > 2$ classes

$$P(y_1 \mid x) + P(y_2 \mid x) + \ldots + P(y_K \mid x) = 1$$

Multinomial Logistic Regression

The probability of everything must still sum to 1

$K > 2$ classes

$$P(y_1 \mid x) + P(y_2 \mid x) + \ldots + P(y_K \mid x) = 1$$

Softmax

- Generalize the sigmoid function: softmax
  - Input: a vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]$ of K arbitrary values
    - each z_i corresponds to weighted sum of features for the Kth class
  - Outputs a probability distribution

$$\textbf{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \qquad 1 \leq i \leq K$$

Multinomial Logistic Regression

The probability of everything must still sum to 1

$K > 2$ classes

$$P(y_1 \mid x) + P(y_2 \mid x) + \ldots + P(y_K \mid x) = 1$$

Softmax

- Generalize the sigmoid function: softmax
  - Input: a vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]$ of K arbitrary values
    - each z_i corresponds to weighted sum of features for the Kth class
  - Outputs a probability distribution

$$\textbf{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \qquad 1 \leq i \leq K$$

$$\textbf{softmax}(\mathbf{z}) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{K} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{K} \exp(z_i)}, \ldots, \frac{\exp(z_K)}{\sum_{i=1}^{K} \exp(z_i)} \right]$$

Multinomial Logistic Regression

The probability of everything must still sum to 1

$K > 2$ classes

$$P(y_1 \mid x) + P(y_2 \mid x) + \ldots + P(y_K \mid x) = 1$$

Softmax

- Generalize the sigmoid function: softmax
  - Input: a vector $\mathbf{z} = [z_1, z_2, \ldots, z_K]$ of K arbitrary values
    - each z_i corresponds to weighted sum of features for the Kth class
  - Outputs a probability distribution

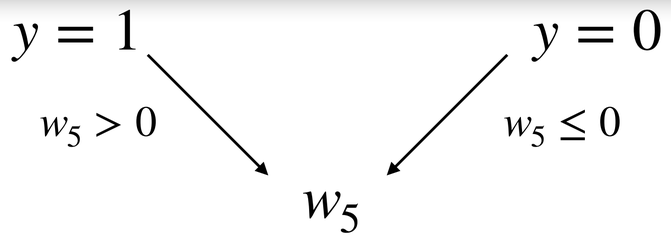
$$\textbf{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)} \qquad 1 \leq i \leq K$$

$$\textbf{softmax}(\mathbf{z}) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{K} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{K} \exp(z_i)}, \ldots, \frac{\exp(z_K)}{\sum_{i=1}^{K} \exp(z_i)} \right]$$

The denominator $\sum_{i=1}^{K} \exp(z_i)$ is used to normalize all the values into probabilities.

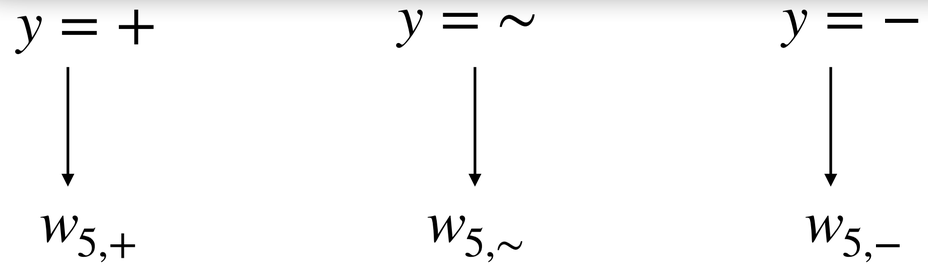
Binary versus Multinomial

Binary Logistic Regression



$$x_5 = \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \qquad w_5 = 3.0$$

Multinomial Logistic Regression



Separate weights for each class

Feature	Definition	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	−5.3

Words, words, words

~~Words, words, words~~

Types, types, types

What do words mean?

A sense or "concept" is the meaning component of a word

"breaking"

Lemmas

- Canonical form
- For example,
  break, breaks, broke, broken a
  nd breaking all share the
  lemma "break"

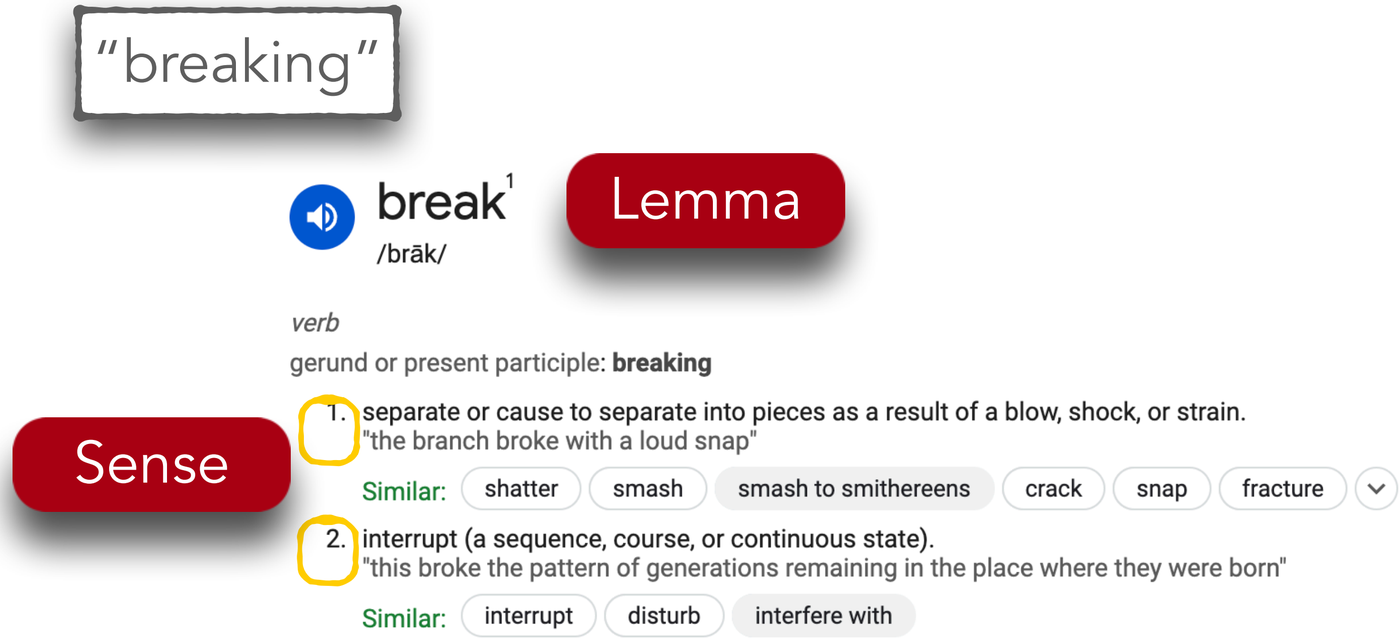
Can be polysemous (have multiple senses)

What do words mean?

A sense or "concept" is the meaning component of a word

Lemmas
- Canonical form
- For example,
  break, breaks, broke, broken a
  nd breaking all share the
  lemma "break"

Can be polysemous (have multiple senses)

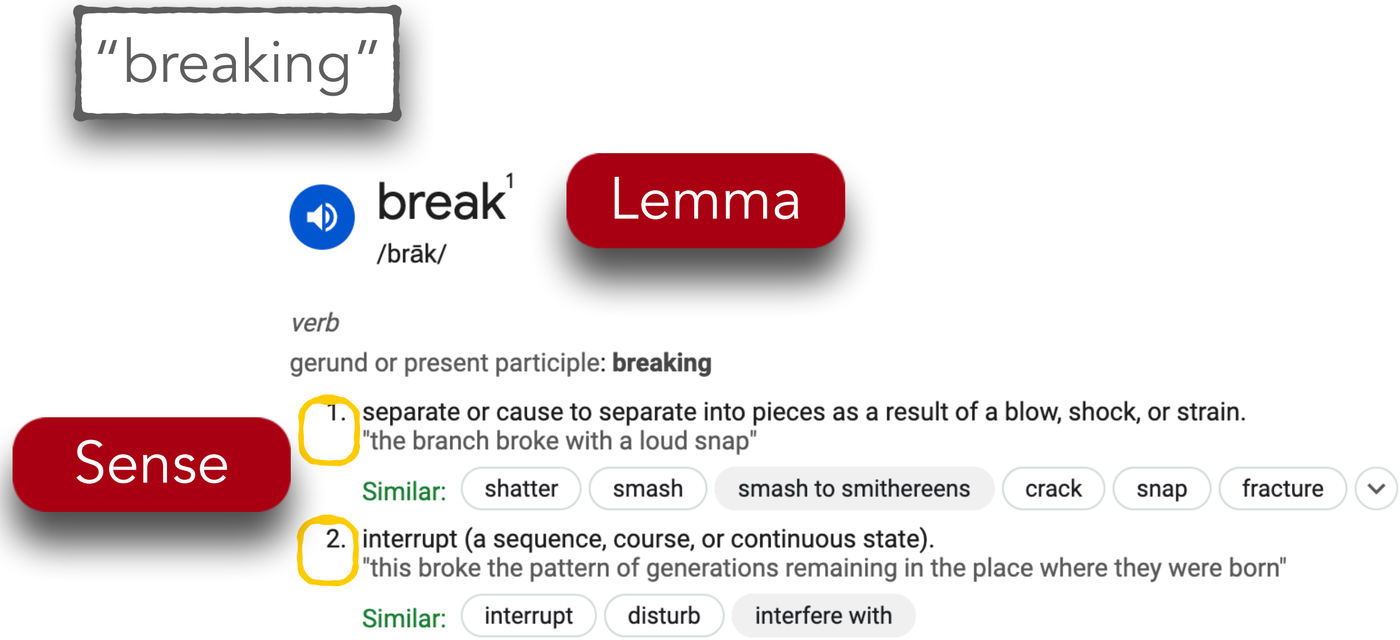


What do words mean?

A sense or "concept" is the meaning component of a word

Lemmas
- Canonical form
- For example,
  break, breaks, broke, broken a
  nd breaking all share the
  lemma "break"

Can be polysemous (have multiple senses)



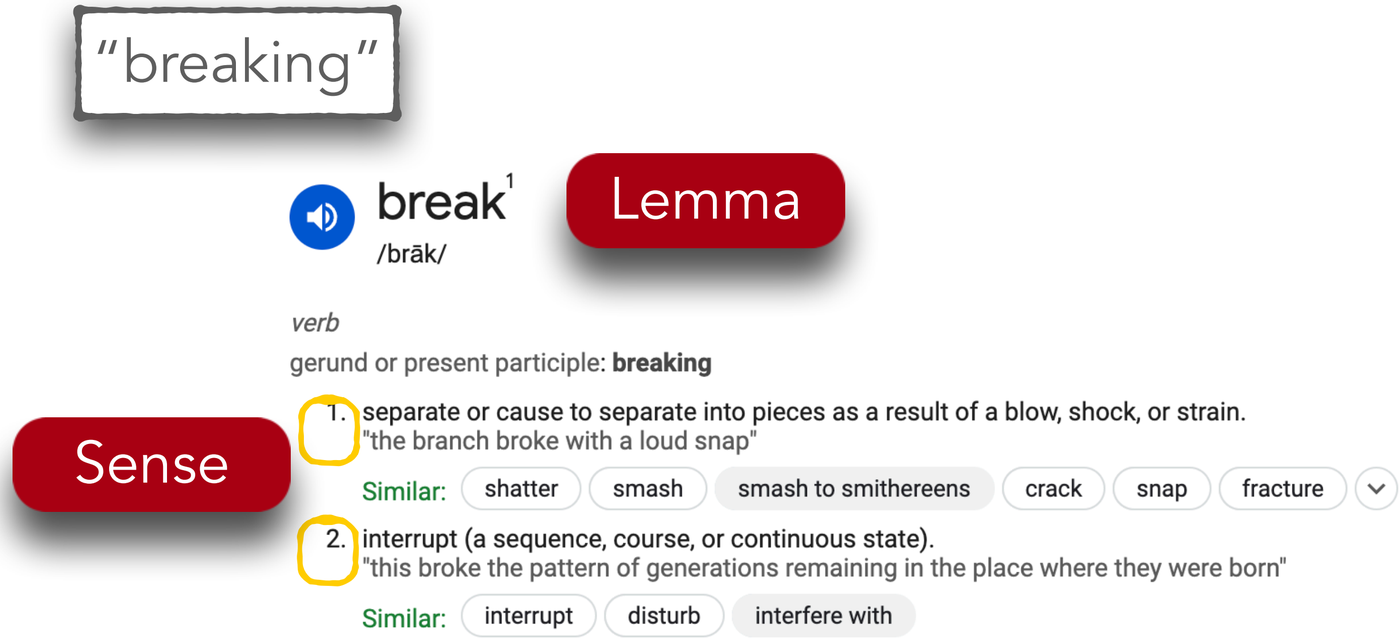
Synonyms: words that have the same meaning in some or all contexts
- e.g. couch / sofa

What do words mean?

A sense or "concept" is the meaning component of a word

Lemmas
- Canonical form
- For example,
  break, breaks, broke, broken a
  nd breaking all share the
  lemma "break"

Can be polysemous (have multiple senses)



Synonyms: words that have the same meaning in some or all contexts
- e.g. couch / sofa

Is perfect synonymy possible

Similarity

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

word1	word2
vanish	disappear
behave	obey
belief	impression
muscle	bone
modest	flexible
hole	agreement

Similarity

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

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

Human assessment of word similarity

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

WordNet


PRINCETON UNIVERSITY
WordNet
A Lexical Database for English

WordNet

- WordNet® is a large lexical database of English

WordNet

- WordNet® is a large lexical database of English
- Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept

WordNet

- WordNet® is a large lexical database of English
- Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept
- Relations between synsets:
  - Super-subordinate relations (hyperonymy, hyponymy or ISA relation)
    - an armchair is a kind of chair, chair is a kind of furniture

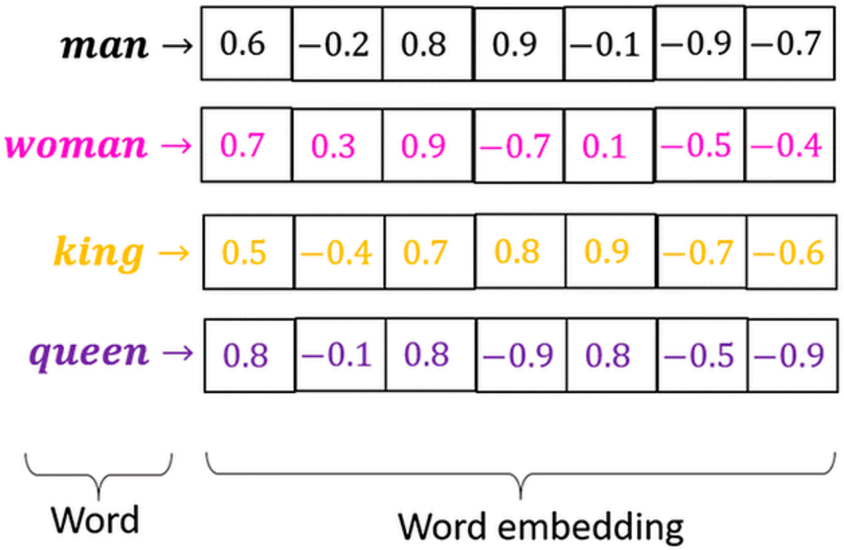
WordNet

- WordNet® is a large lexical database of English
- Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept
- Relations between synsets:
  - Super-subordinate relations (hyperonymy, hyponymy or ISA relation)
    - an armchair is a kind of chair, chair is a kind of furniture
  - Meronymy (part-of)
    - chair has legs

WordNet

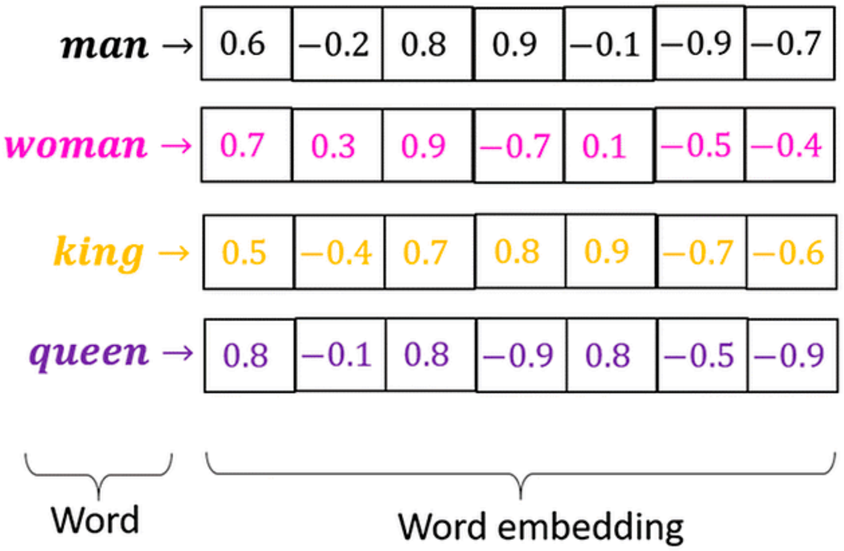
- WordNet® is a large lexical database of English
- Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (synsets), each expressing a distinct concept
- Relations between synsets:
  - Super-subordinate relations (hyperonymy, hyponymy or ISA relation)
    - an armchair is a kind of chair, chair is a kind of furniture
  - Meronymy (part-of)
    - chair has legs
  - Antonymy

Words as Vectors



Words as Vectors

In NLP, we commonly represent word types with vectors!

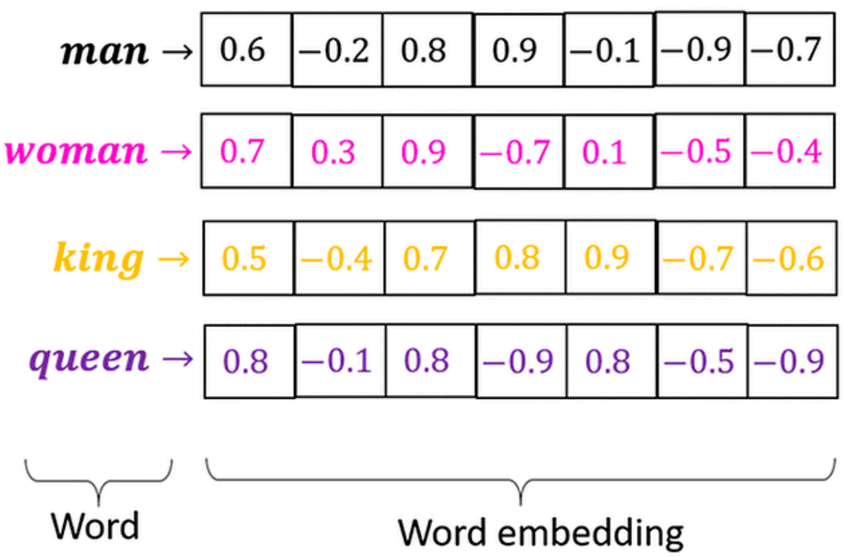


Words as Vectors

In NLP, we commonly represent word types with vectors!

- Very useful in capturing similarity between words, and other forms of lexical semantics (e.g. synonymy, hypernyms, antonymy)

Why?

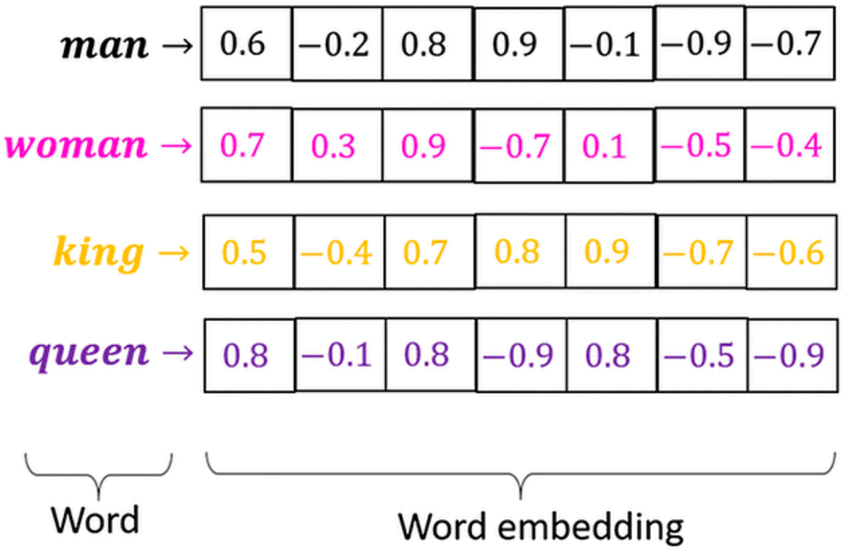


Words as Vectors

In NLP, we commonly represent word types with vectors!

- Very useful in capturing similarity between words, and other forms of lexical semantics (e.g. synonymy, hypernyms, antonymy)
- Computing the similarity between two words (or phrases, or documents) is extremely useful for many NLP tasks
  - Q: How **tall** is Mount Everest?
  - A: The official **height** of Mount Everest is 29029 ft

Why?



"You shall know a word by the company it keeps."

- Firth (1957)

Word Meaning via Language Use

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word “usage”: word environments / contexts

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word “usage”: word environments / contexts
    - Neighboring words or grammatical environments

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word “usage”: word environments / contexts
    - Neighboring words or grammatical environments

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word “usage”: word environments / contexts
    - Neighboring words or grammatical environments

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word “usage”: word environments / contexts
    - Neighboring words or grammatical environments
- Intuitions: Zellig Harris (1954):
  - “oculist and eye-doctor … occur in almost the same environments”

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word "usage": word environments / contexts
    - Neighboring words or grammatical environments
- Intuitions: Zellig Harris (1954):
  - "oculist and eye-doctor … occur in almost the same environments"
  - "If A and B have almost identical environments we say that they are synonyms."

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Word Meaning via Language Use

- The meaning of a word can be given by its distribution in language usage
  - Word "usage": word environments / contexts
    - Neighboring words or grammatical environments
- Intuitions: Zellig Harris (1954):
  - "oculist and eye-doctor … occur in almost the same environments"
  - "If A and B have almost identical environments we say that they are synonyms."

A bottle of tesgüino is on the table
Everybody likes tesgüino
Tesgüino makes you drunk
We make tesgüino out of corn.

Two words are similar if they have similar word contexts


You think you just fell
out of a coconut tree?


YOU THINK YOU JUST FELL OUT
OF A
COCONUT
TREE?
YOU EXIST IN THE CONTEXT OF ALL IN WHICH
YOU LIVE AND WHAT CAME BEFORE YOU.

Words exist in the context of all that came before…


"In Springfield,
they're eating the dogs,
the people that came in,
they're eating the cats...
They're eating...
they're eating the pets of
the people that live there"
Trump 09/10/2024


FOX NEWS channel
FOX NEWS DEMOCRACY '24
LIVE abc NEWS
2024 PRESIDENTIAL DEBATE

When seen in atypical contexts, words may be associated with high surprisal

When seen in atypical contexts, words may be associated with high surprisal

Conspiracy Theories Warning

When seen in atypical contexts, words may be associated with high surprisal

Conspiracy Theories Warning

Word Embeddings

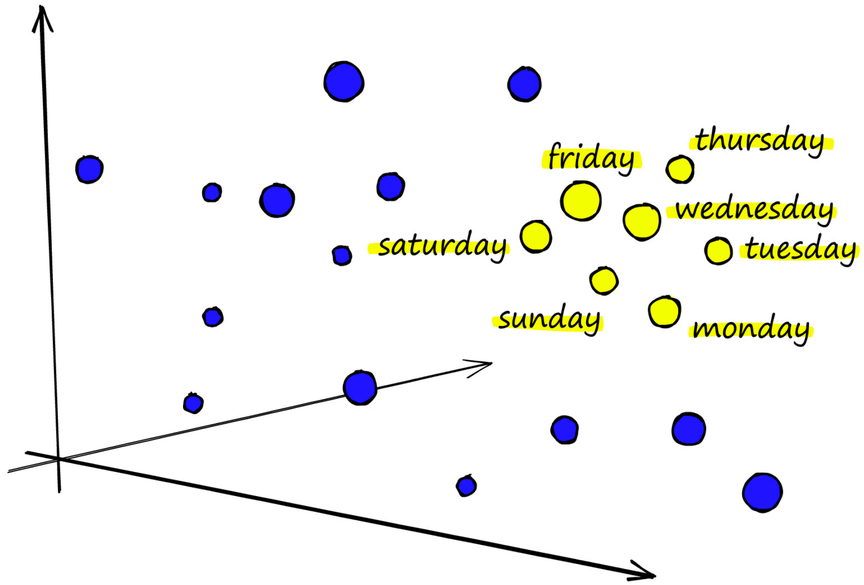


Image Credit: Pinecone

Word Embeddings

- Represent a word as a point in a multidimensional semantic space

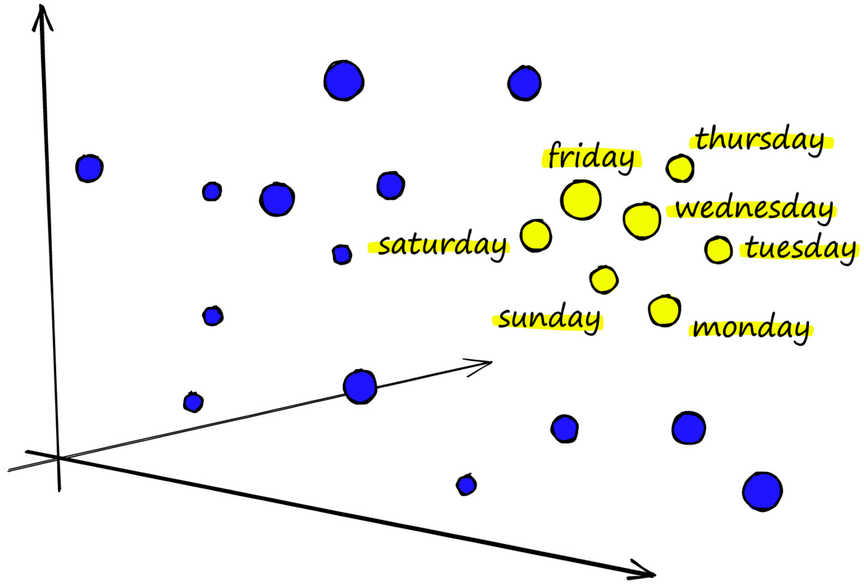


Image Credit: Pinecone

Word Embeddings

- Represent a word as a point in a multidimensional semantic space
  - Space itself constructed from distribution of word neighbors

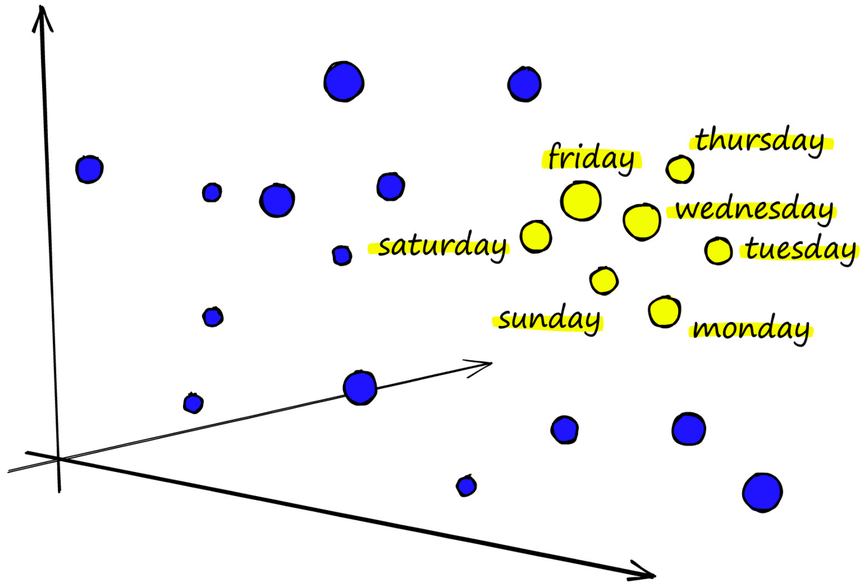


Image Credit: Pinecone

Word Embeddings

Vector Semantics

- Represent a word as a point in a multidimensional semantic space
  - Space itself constructed from distribution of word neighbors

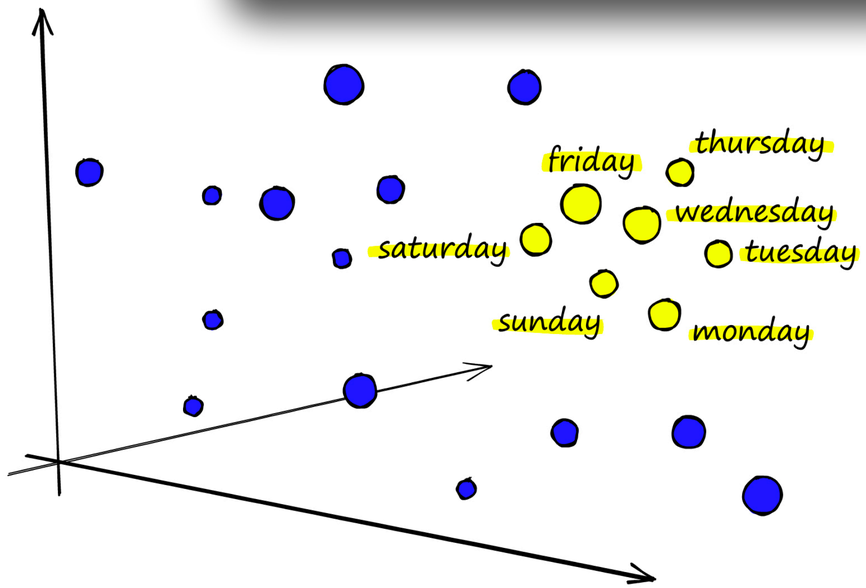


Image Credit: Pinecone

Word Embeddings

Vector Semantics

- Represent a word as a point in a multidimensional semantic space
  - Space itself constructed from distribution of word neighbors
- Called an "embedding" because it's embedded into a space

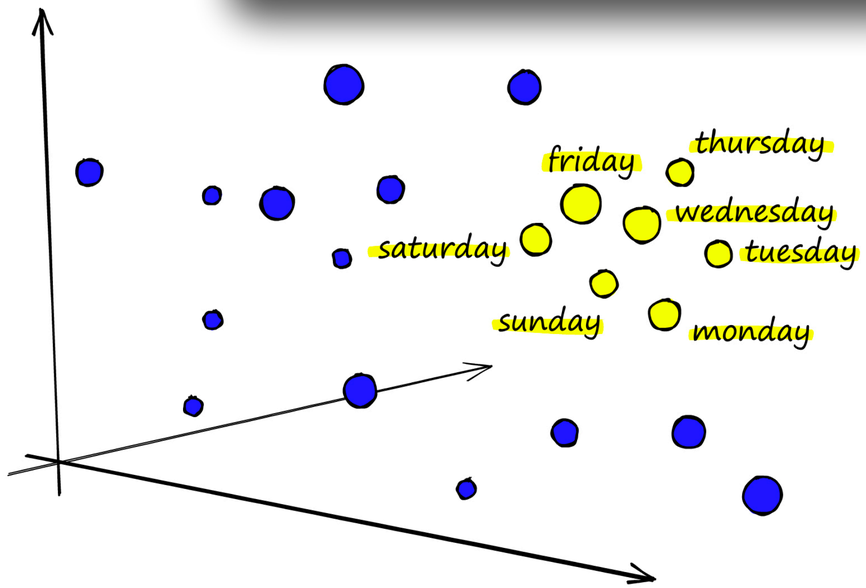


Image Credit: Pinecone

Word Embeddings

Vector Semantics

- Represent a word as a point in a multidimensional semantic space
  - Space itself constructed from distribution of word neighbors
- Called an "embedding" because it's embedded into a space
- Fine-grained model of meaning for **similarity**

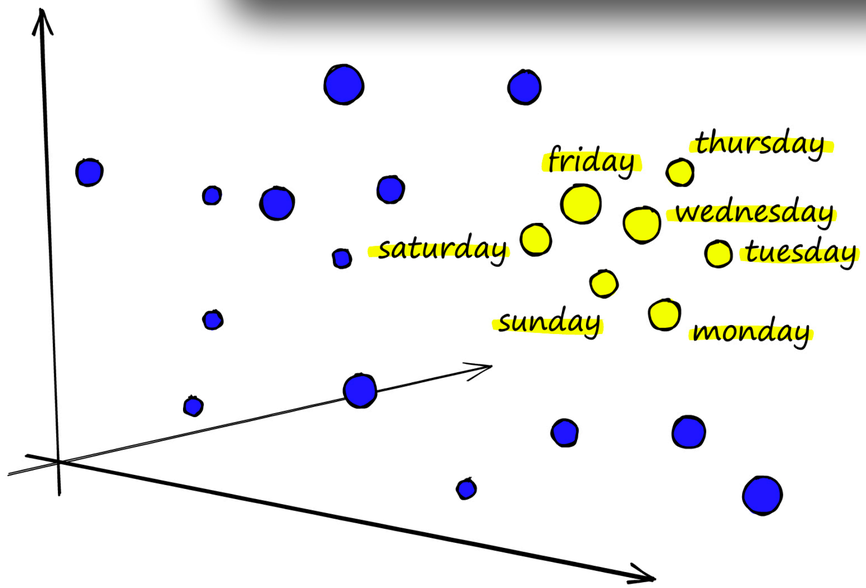
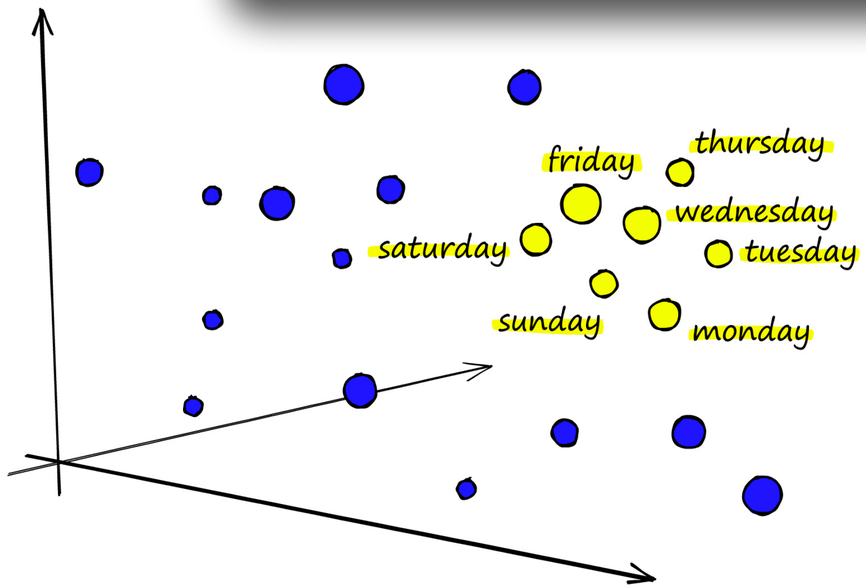


Image Credit: Pinecone

Word Embeddings

Vector Semantics

- Represent a word as a point in a multidimensional semantic space
  - Space itself constructed from distribution of word neighbors
- Called an "embedding" because it's embedded into a space
- Fine-grained model of meaning for **similarity**



Every modern NLP algorithm uses embeddings as the representation of word meaning

Image Credit: Pinecone

荃者所以在鱼，得鱼而忘荃

言者所以在意，得意而忘言

"Nets are for fish; Once you get the fish, you can forget the net.

Words are for meaning; Once you get the meaning, you can forget the words"

– *Zhuangzi*

庄子

Cosine Similarity for Word Similarity

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

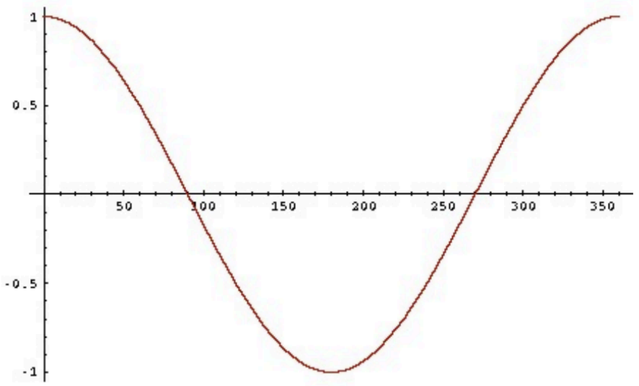
Cosine Similarity for Word Similarity

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

-1: vectors point in opposite directions

+1: vectors point in same directions

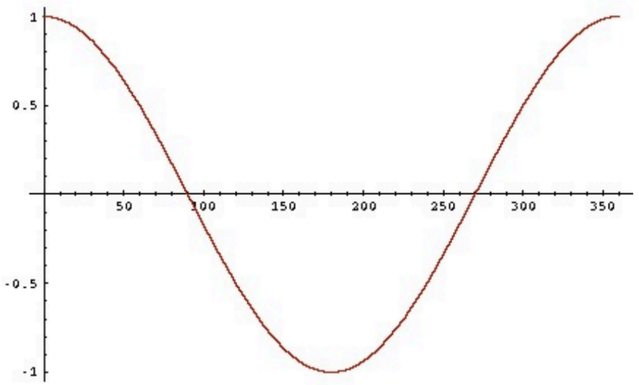
0: vectors are orthogonal



Cosine Similarity for Word Similarity

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

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

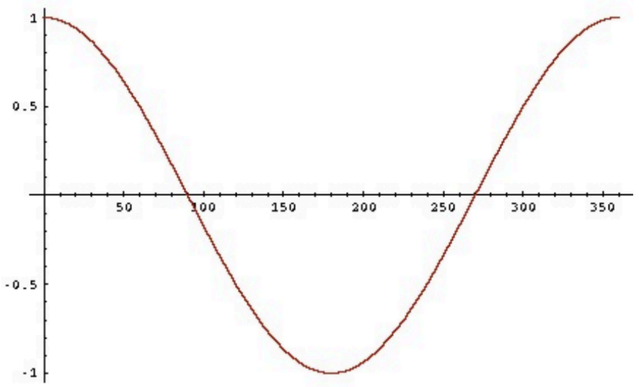


Greater the cosine, more similar the words

Cosine Similarity for Word Similarity

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

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



We do not care about the magnitude of the embeddings, just the angle between them

Greater the cosine, more similar the words

n-grams as One-hot Vectors

Unigram Vectors: Represent each word as a vector of zeros with a single 1 identifying the index of the word

n-grams as One-hot Vectors

vocabulary
i
hate
love
the
movie
film

movie = <0, 0, 0, 0, 1, 0>
film = <0, 0, 0, 0, 0, 1>

Unigram Vectors: Represent each word as a vector of zeros with a single 1 identifying the index of the word

One hot vector

n-grams as One-hot Vectors

vocabulary
i
hate
love
the
movie
film

movie = <0, 0, 0, 0, 1, 0>
film = <0, 0, 0, 0, 0, 1>

One hot vector

Unigram Vectors: Represent each word as a vector of zeros with a single 1 identifying the index of the word

Dot product is zero! These vectors are orthogonal

n-grams as One-hot Vectors

vocabulary
i
hate
love
the
movie
film

movie = <0, 0, 0, 0, 1, 0>
film = <0, 0, 0, 0, 0, 1>

One hot vector

Unigram Vectors: Represent each word as a vector of zeros with a single 1 identifying the index of the word

How can we compute a vector representation such that the dot product correlates with word similarity?

Dot product is zero! These vectors are orthogonal

Let us consider a collection of documents and count how frequently a word (**term**) appears in each. A document could be a play or a Wikipedia article. In general, documents can be anything; we often call each paragraph a document!

Term-document matrix

Let us consider a collection of documents and count how frequently a word (**term**) appears in each. A document could be a play or a Wikipedia article. In general, documents can be anything; we often call each paragraph a document!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Term-document matrix

Let us consider a collection of documents and count how frequently a word (**term**) appears in each. A document could be a play or a Wikipedia article. In general, documents can be anything; we often call each paragraph a document!

Each **document** is represented by a vector of words

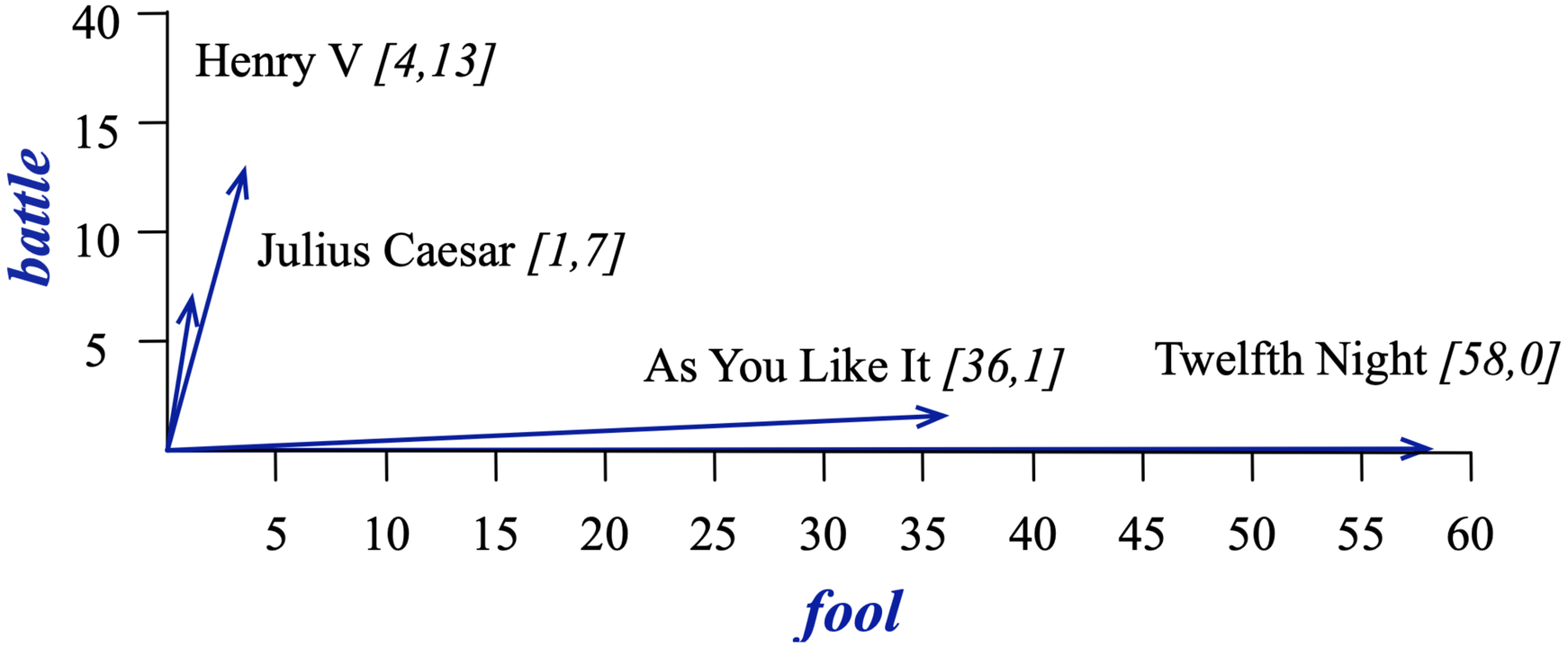
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Visualizing document vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

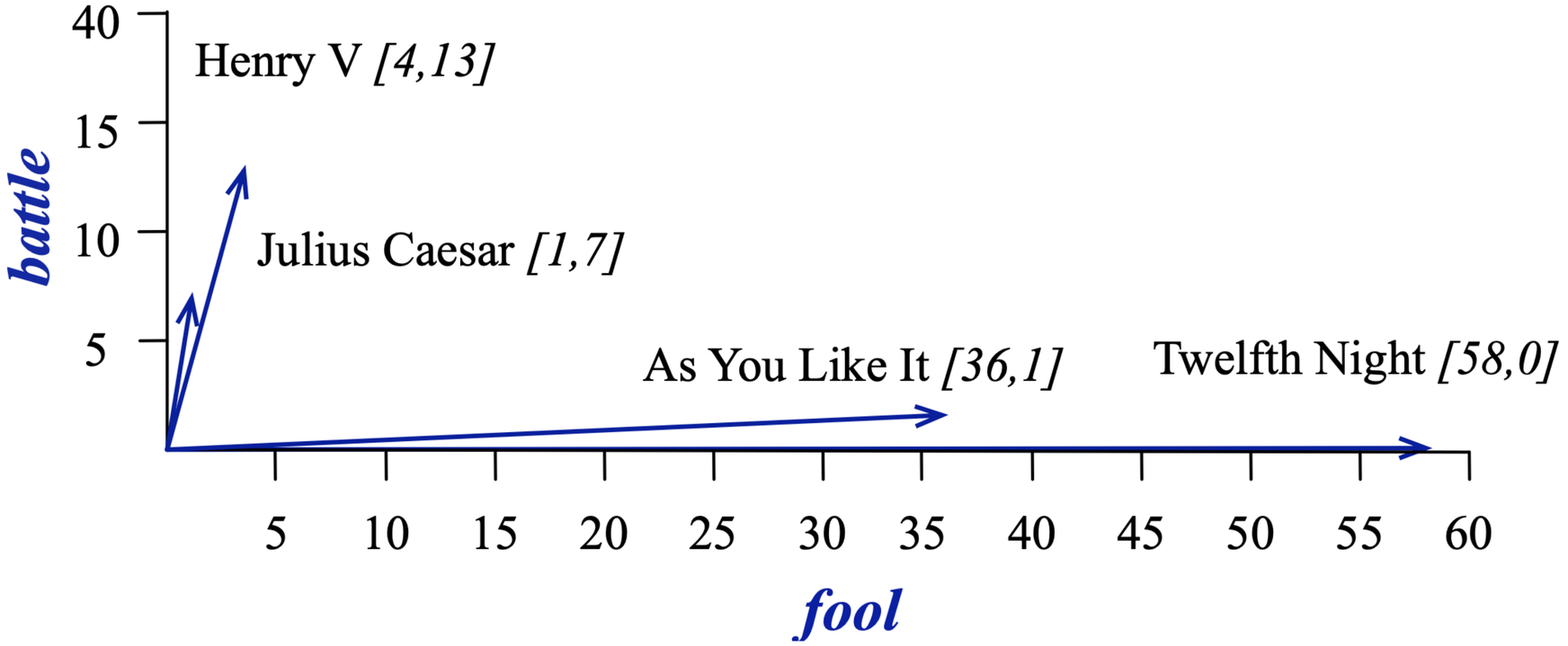
Visualizing document vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3



Visualizing document vectors

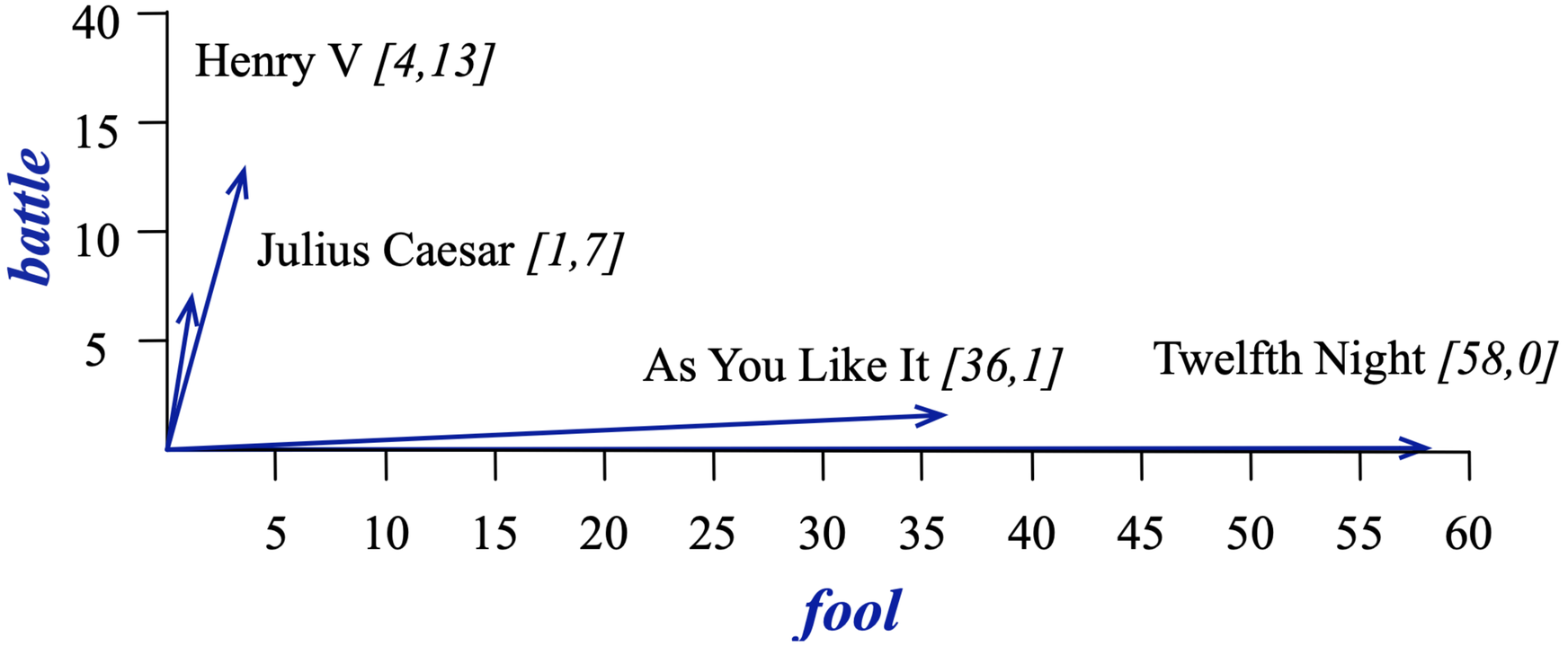
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3



- Vectors are similar for the two comedies

Visualizing document vectors

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3



- Vectors are similar for the two comedies
- Comedies are different from the other two (tragedies)
  - More fools, less battle

Words as vectors in a co-occurrence matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Words as vectors in a co-occurrence matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

"Battle" is the kind of word that appears in Julius Caesar and Henry V

"Fool" is the kind of word that appears in As You Like It and Twelfth Night

Words as vectors in a co-occurrence matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

"Battle" is the kind of word that appears in Julius Caesar and Henry V

"Fool" is the kind of word that appears in As You Like It and Twelfth Night

Number of dimensions?

Word-word co-occurrence matrix

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

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

Word-word co-occurrence matrix

Context Window

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

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

Word-word co-occurrence matrix

Context Window

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

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

	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

Word-word co-occurrence matrix

Context Window

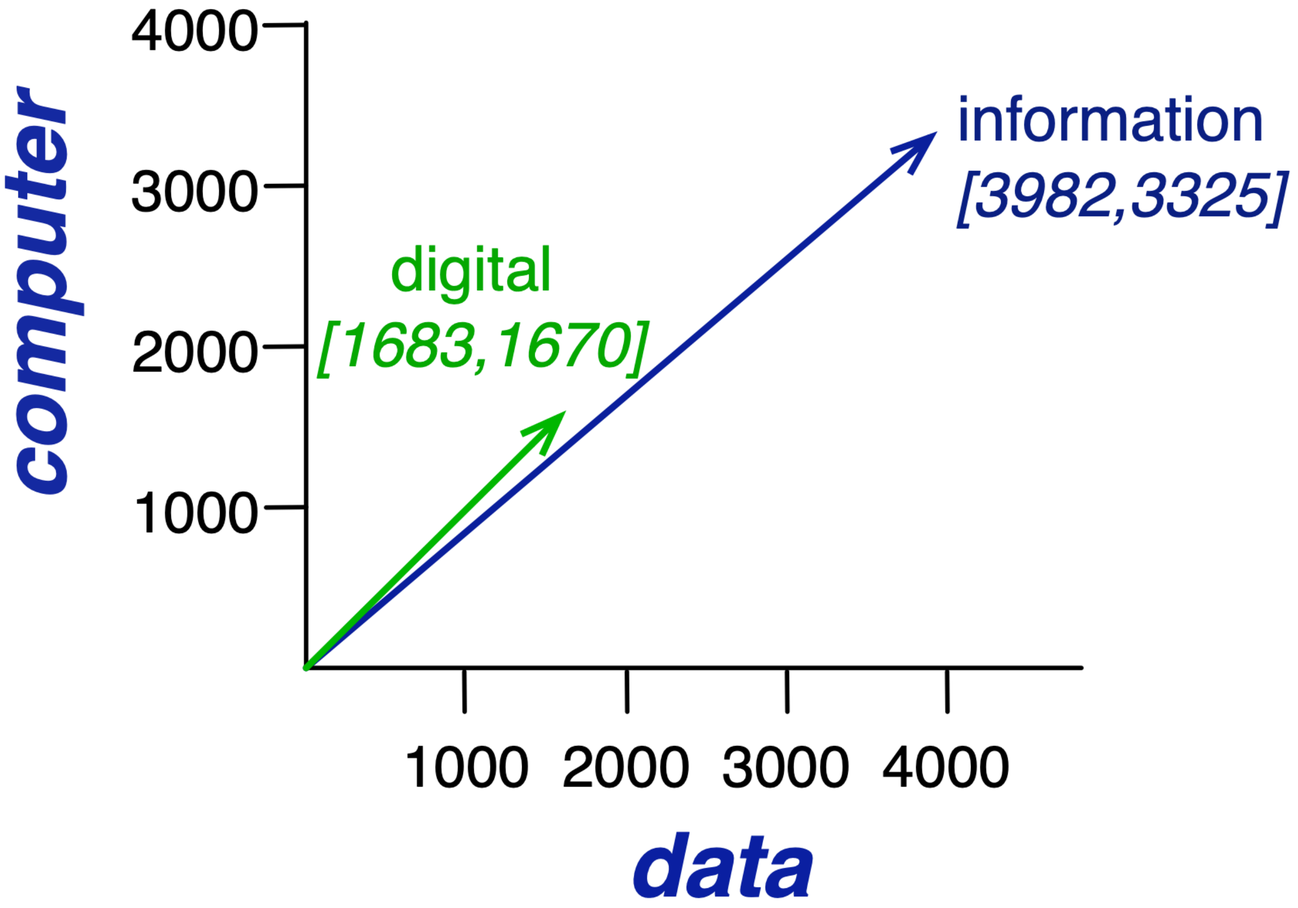
is traditionally followed by	**cherry**	pie, a traditional dessert
often mixed, such as	**strawberry**	rhubarb pie. Apple pie
computer peripherals and personal	**digital**	assistants. These devices usually
a computer. This includes	**information**	available on the internet

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

Words, not documents

	aardvark	...	**computer**	**data**	**result**	**pie**	**sugar**	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	...
strawberry	0	...	0	0	1	60	19	...
digital	0	...	1670	1683	85	5	4	...
information	0	...	3325	3982	378	5	13	...



Raw frequencies though…

Raw frequencies though…

- …are a bad representation!

Raw frequencies though…

- …are a bad representation!
- The co-occurrence matrices we have seen represent each cell by word frequencies

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

Raw frequencies though…

- …are a bad representation!
- The co-occurrence matrices we have seen represent each cell by word frequencies
- Frequency is clearly useful; if sugar appears a lot near apricot, that's useful information

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

Raw frequencies though…

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

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

Raw frequencies though…

- …are a bad representation!
- The co-occurrence matrices we have seen represent each cell by word frequencies
- Frequency is clearly useful; if sugar appears a lot near apricot, that's useful information
- But overly frequent words like the, it, or they are not very informative about the context
- It's a paradox! How can we balance these two conflicting constraints?

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

Raw frequencies though…

- …are a bad representation!
- The co-occurrence matrices we have seen represent each cell by word frequencies
- Frequency is clearly useful; if sugar appears a lot near apricot, that's useful information
- But overly frequent words like the, it, or they are not very informative about the context
- It's a paradox! How can we balance these two conflicting constraints?

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

Need some form of weighting!

Pointwise Mutual Information (PMI)

Church & Hanks 1989

Pointwise Mutual Information (PMI)

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

- PMI ranges from $-\infty$ to $+\infty$

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

- PMI ranges from $-\infty$ to $+\infty$
  - Negative values are problematic: words are co-occurring less than we expect by chance

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

- PMI ranges from $-\infty$ to $+\infty$
  - Negative values are problematic: words are co-occurring less than we expect by chance
  - Only reliable under an enormous corpora
    - Imagine w_1 and w_2 whose probability of occurrence is each 10^{-6}
    - Hard to be sure $P(w_1, w_2)$ is significantly different than 10^{-12}

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

- PMI ranges from $-\infty$ to $+\infty$
  - Negative values are problematic: words are co-occurring less than we expect by chance
  - Only reliable under an enormous corpora
    - Imagine w_1 and w_2 whose probability of occurrence is each 10^{-6}
    - Hard to be sure $P(w_1, w_2)$ is significantly different than 10^{-12}
  - So we just replace negative PMI values by 0

Church & Hanks 1989

Pointwise Mutual Information (PMI)

$$PMI(w_1, w_2) = \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

- **PMI between two words:**
  - Do words w_1 and w_2 co-occur more than if they were independent?

- PMI ranges from $-\infty$ to $+\infty$
  - Negative values are problematic: words are co-occurring less than we expect by chance
  - Only reliable under an enormous corpora
    - Imagine w_1 and w_2 whose probability of occurrence is each 10^{-6}
    - Hard to be sure $P(w_1, w_2)$ is significantly different than 10^{-12}
  - So we just replace negative PMI values by 0

- **Positive PMI**

$$PPMI(w_1, w_2) = \max\left(0, \log \frac{P(w_1, w_2)}{P(w_1)P(w_2)}\right)$$

Church & Hanks 1989

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Frequency $f(w_i, c_j)$ or f_{ij} is the # times w_i occurs in context c_j

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Frequency $f(w_i, c_j)$ or f_{ij} is the # times w_i occurs in context c_j

$$P_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W} \sum_{j=1}^{C} f_{i,j}}$$

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Frequency $f(w_i, c_j)$ or f_{ij} is the # times w_i occurs in context c_j

$$P_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{i,j}} \qquad P_i = \sum_{j=1}^{C} P_{ij}$$

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Frequency $f(w_i, c_j)$ or f_{ij} is the # times w_i occurs in context c_j

$$P_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{i,j}} \qquad P_i = \sum_{j=1}^{C} P_{ij} \qquad P_j = \sum_{i=1}^{W} P_{ij}$$

Computing PPMI on a term-context matrix

Context c

Term w

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

Frequency $f(w_i, c_j)$ or f_{ij} is the # times w_i occurs in context c_j

$$P_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{i,j}} \qquad P_i = \sum_{j=1}^{C} P_{ij} \qquad P_j = \sum_{i=1}^{W} P_{ij}$$

$$PPMI(w_i, c_j) = PPMI_{i,j} = \max\left(0, \log \frac{P_{ij}}{P_i P_j}\right)$$

The problem…

The problem…

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

The problem…

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

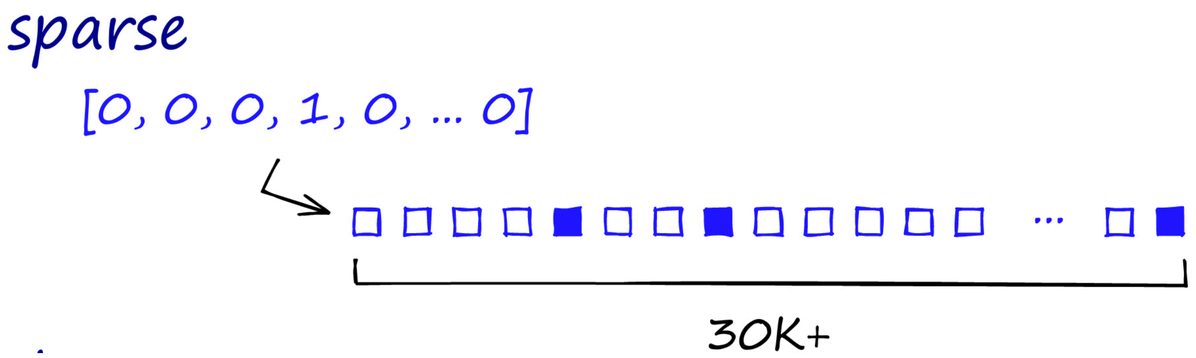


Image Credit: Pinecone

The problem…

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

- Alternative: learn vectors which are
  - short (length 50-1000)
  - dense (most elements are non-zero)

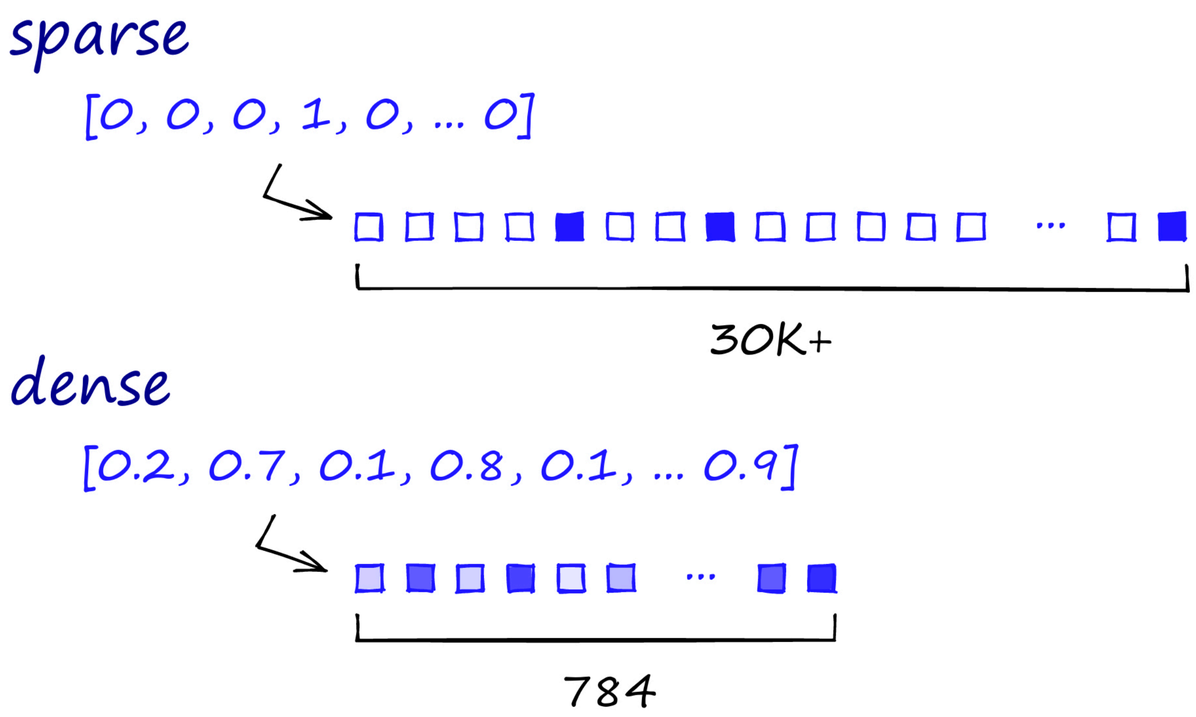


Image Credit: Pinecone

Sparse vs. Dense Vectors

Sparse vs. Dense Vectors

- Why dense vectors?

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts
  - Dense vectors may do better at capturing synonymy:

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts
  - Dense vectors may do better at capturing synonymy:
    - car and automobile are synonyms; but are distinct dimensions

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts
  - Dense vectors may do better at capturing synonymy:
    - car and automobile are synonyms; but are distinct dimensions
    - a word with car as a neighbor and a word with automobile as a neighbor should be similar, but aren't

Sparse vs. Dense Vectors

- Why dense vectors?
  - Memory efficiency is not a problem for sparse vectors…
  - Short vectors may be easier to use as features in machine learning (fewer weights to tune)
  - Dense vectors may generalize better than explicit counts
  - Dense vectors may do better at capturing synonymy:
    - car and automobile are synonyms; but are distinct dimensions
    - a word with car as a neighbor and a word with automobile as a neighbor should be similar, but aren't
  - In practice, they work better

How to obtain dense vectors?

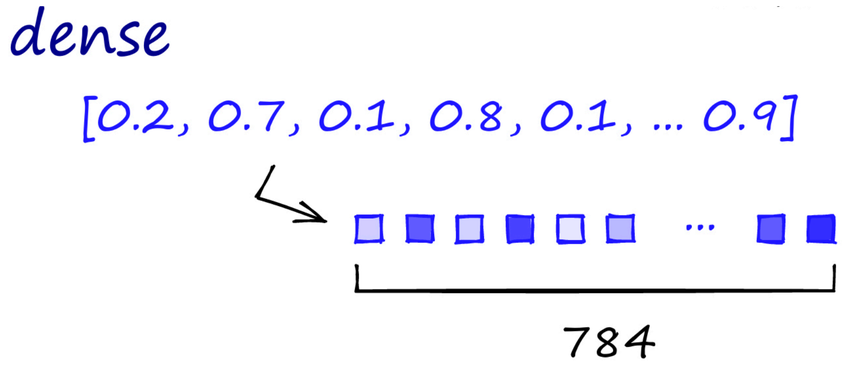


Image Credit: Pinecone

How to obtain dense vectors?

"Neural Language Model"-inspired models

- Word2vec (skipgram, CBOW), GloVe

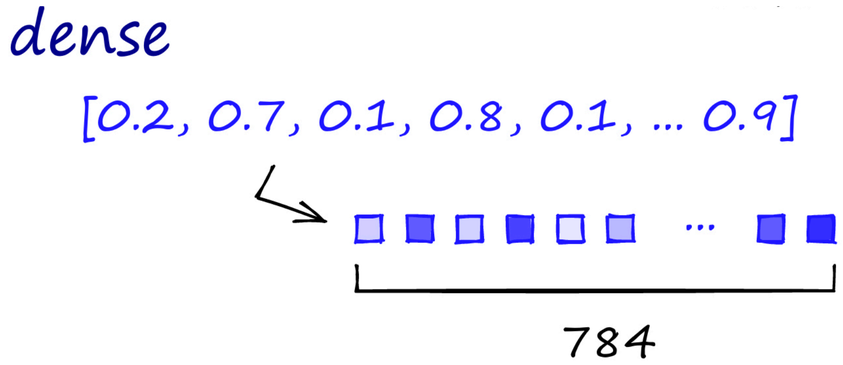


Image Credit: Pinecone

How to obtain dense vectors?

"Neural Language Model"-inspired models

- Word2vec (skipgram, CBOW), GloVe

Singular Value Decomposition (SVD)

- Special case: Latent Semantic Analysis (LSA)

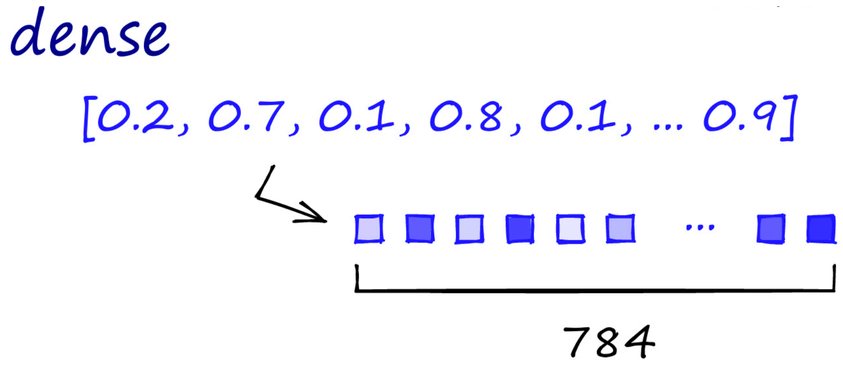


Image Credit: Pinecone

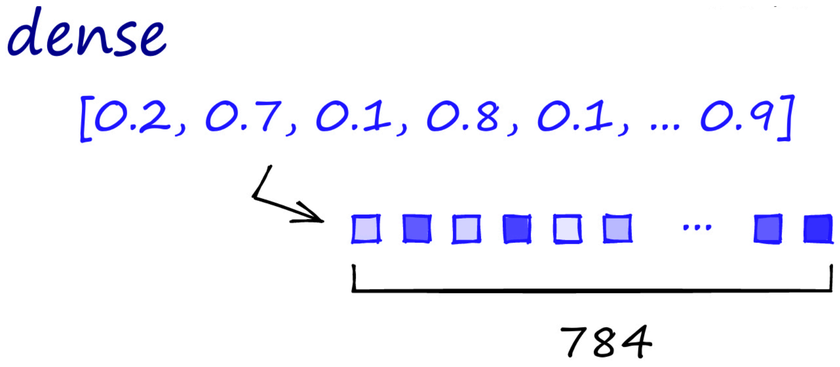
How to obtain dense vectors?

"Neural Language Model"-inspired models

- Word2vec (skipgram, CBOW), GloVe

Singular Value Decomposition (SVD)

- Special case: Latent Semantic Analysis (LSA)



Alternatives to "static word type embeddings":

- Contextual Embeddings (LLM word embeddings)
  - Compute distinct embeddings for a word in its context
  - Separate embeddings for each token of a word

Image Credit: Pinecone

Lecture Outline

- Announcements
- Recap: Multinomial LR
- Recap: Lexical Semantics
- word2vec
  - Classification
  - Learning
- GloVe
- Properties and Evaluation of Word Embeddings

word2vec

word2vec

Mikolov et al., ICLR 2013. Efficient estimation of word representations in vector space.

Mikolov et al., NeurIPS 2013. Distributed representations of words and phrases and their compositionality.

word2vec

- Short, dense vector or embedding

Mikolov et al., ICLR 2013. Efficient estimation of word representations in vector space.

Mikolov et al., NeurIPS 2013. Distributed representations of words and phrases and their compositionality.

word2vec

- Short, dense vector or embedding
- Static embeddings
  - One embedding per word type
  - Does not change with context change

What happens to the problem of polysemy?

Mikolov et al., ICLR 2013. Efficient estimation of word representations in vector space.

Mikolov et al., NeurIPS 2013. Distributed representations of words and phrases and their compositionality.

word2vec

- Short, dense vector or embedding
- Static embeddings
  - One embedding per word type
  - Does not change with context change
- Two algorithms for computing:
  - Skip-Gram with Negative Sampling or SGNS
  - CBOW or continuous bag of words
  - But we will study a slightly different version…

What happens to the problem of polysemy?

Mikolov et al., ICLR 2013. Efficient estimation of word representations in vector space.

Mikolov et al., NeurIPS 2013. Distributed representations of words and phrases and their compositionality.

word2vec

- Short, dense vector or embedding
- Static embeddings
  - One embedding per word type
  - Does not change with context change
- Two algorithms for computing:
  - Skip-Gram with Negative Sampling or SGNS
  - CBOW or continuous bag of words
  - But we will study a slightly different version…
- Efficient training
- Easily available to download and plug in

What happens to the problem of polysemy?

Mikolov et al., ICLR 2013. Efficient estimation of word representations in vector space.

Mikolov et al., NeurIPS 2013. Distributed representations of words and phrases and their compositionality.

word2vec : Intuition

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

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. “cherry”

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. “cherry”

- Train a classifier on a binary prediction task:

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. “cherry”

- Train a classifier on a binary prediction task:
  - Is w likely to show up near “cherry”?

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. "cherry"

- Train a classifier on a binary prediction task:
  - Is w likely to show up near "cherry"?

What is $\mathbf{x}$? What is y?

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. "cherry"

- Train a classifier on a binary prediction task:
  - Is w likely to show up near "cherry"?
- We don't actually care about this task!!!

What is $\mathbf{x}$? What is y?

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. "cherry"

- Train a classifier on a binary prediction task:
  - Is w likely to show up near "cherry"?
- We don't actually care about this task!!!
  - But we'll take the learned classifier weights as the word embeddings

What is $\mathbf{x}$? What is y?

word2vec : Intuition

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

Instead of counting how often each word w occurs near another, e.g. "cherry"

- Train a classifier on a binary prediction task:
  - Is w likely to show up near "cherry"?
- We don't actually care about this task!!!
  - But we'll take the learned classifier weights as the word embeddings

What is $\mathbf{x}$? What is y?

Word embedding itself is the learned parameter!

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

One missing piece: where to get the (x, y) pairs from?

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

One missing piece: where to get the (x, y) pairs from?

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

- A word c that occurs near "cherry" in the corpus acts as the gold "correct answer" for supervised learning

word2vec: Self-supervision

One missing piece: where to get the (x, y) pairs from?

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

- A word c that occurs near "cherry" in the corpus acts as the gold "correct answer" for supervised learning

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

word2vec: Self-supervision

One missing piece: where to get the (x, y) pairs from?

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

- A word c that occurs near "cherry" in the corpus acts as the gold "correct answer" for supervised learning
- No need for human labels!

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

word2vec: Self-supervision

One missing piece: where to get the (x, y) pairs from?

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

- A word c that occurs near "cherry" in the corpus acts as the gold "correct answer" for supervised learning
- No need for human labels!

What about incorrect labels?

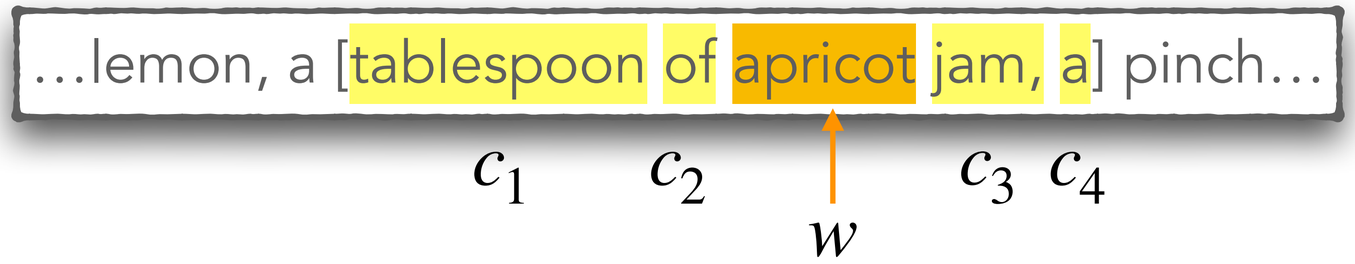
Bengio et al. (2003); Collobert et al. (2011)

word2vec: Goal

Assume a +/- 2 word window, given training sentence:

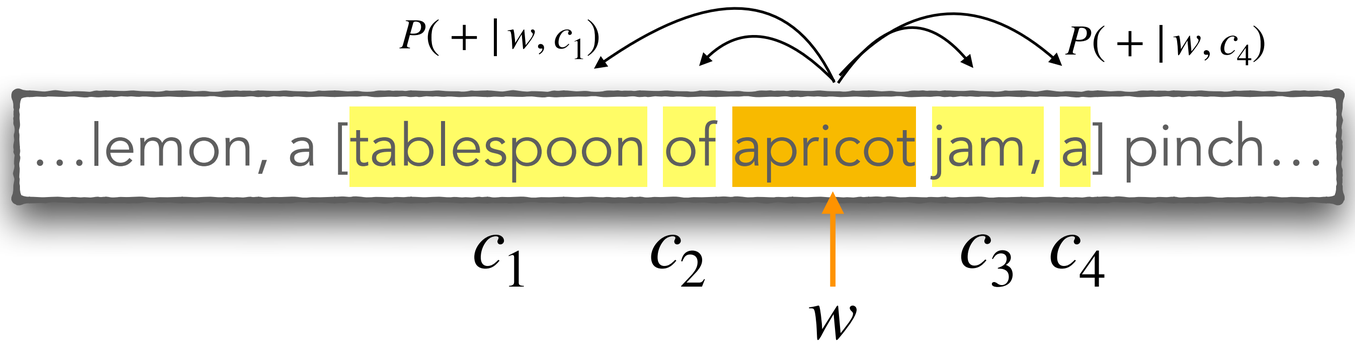
word2vec: Goal

Assume a +/- 2 word window, given training sentence:



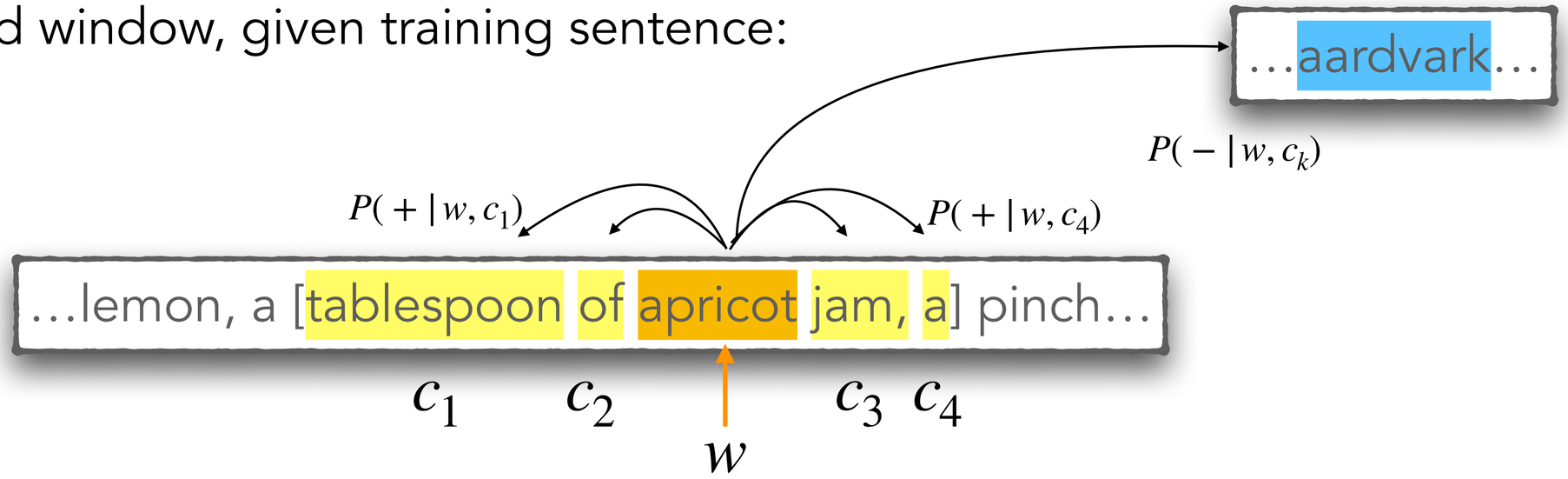
word2vec: Goal

Assume a +/- 2 word window, given training sentence:



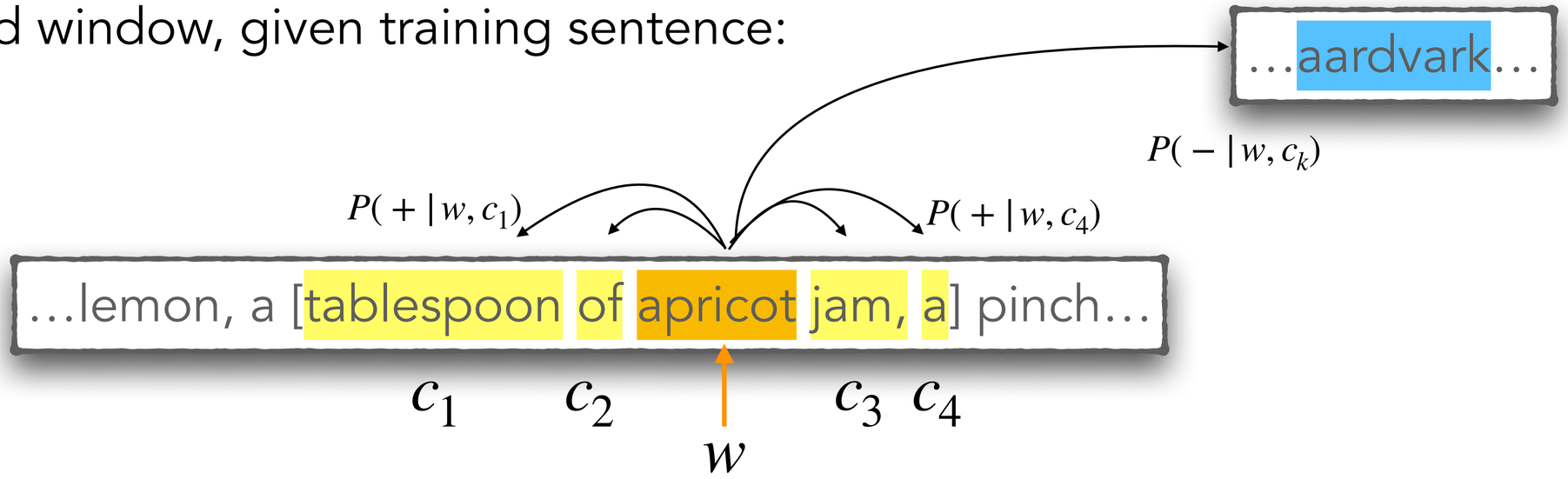
word2vec: Goal

Assume a +/- 2 word window, given training sentence:



word2vec: Goal

Assume a +/- 2 word window, given training sentence:



Goal: train a classifier that is given a candidate (word, context) pair:

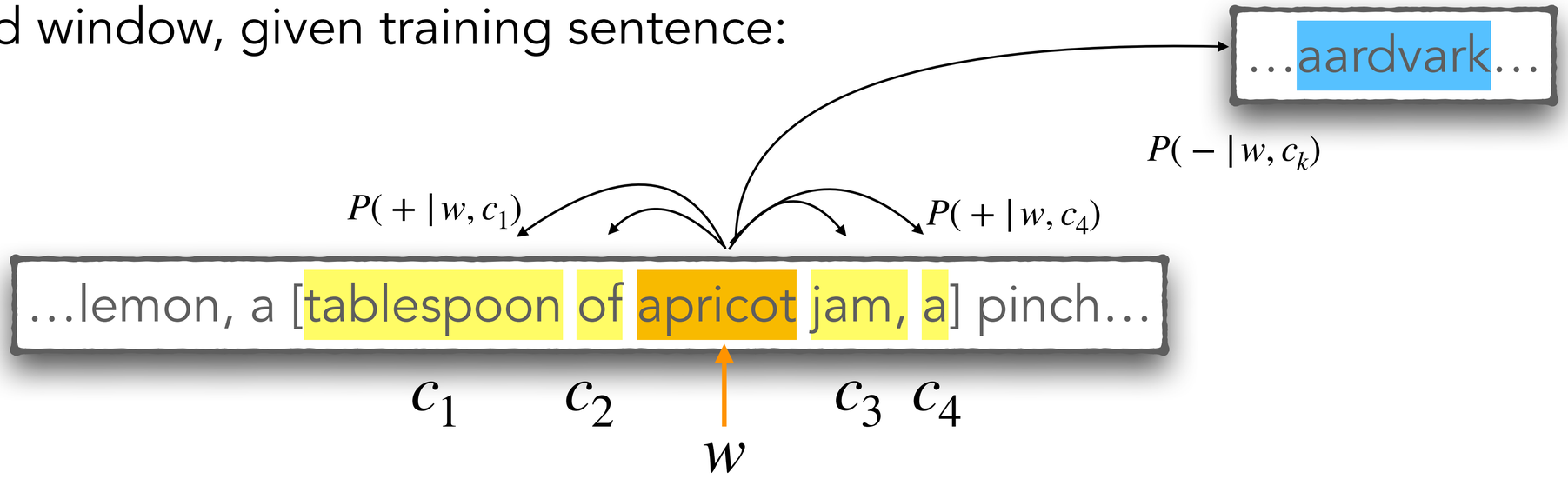
(apricot, jam)

(apricot, aardvark)

…

word2vec: Goal

Assume a +/- 2 word window, given training sentence:



Goal: train a classifier that is given a candidate (word, context) pair:

(apricot, jam)

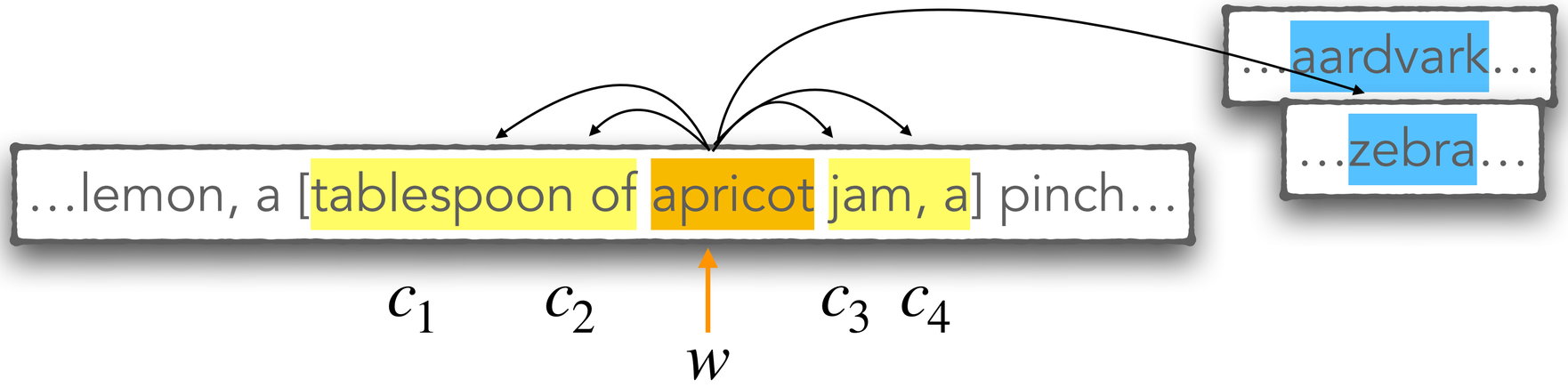
(apricot, aardvark)

…

And assigns each pair a probability:

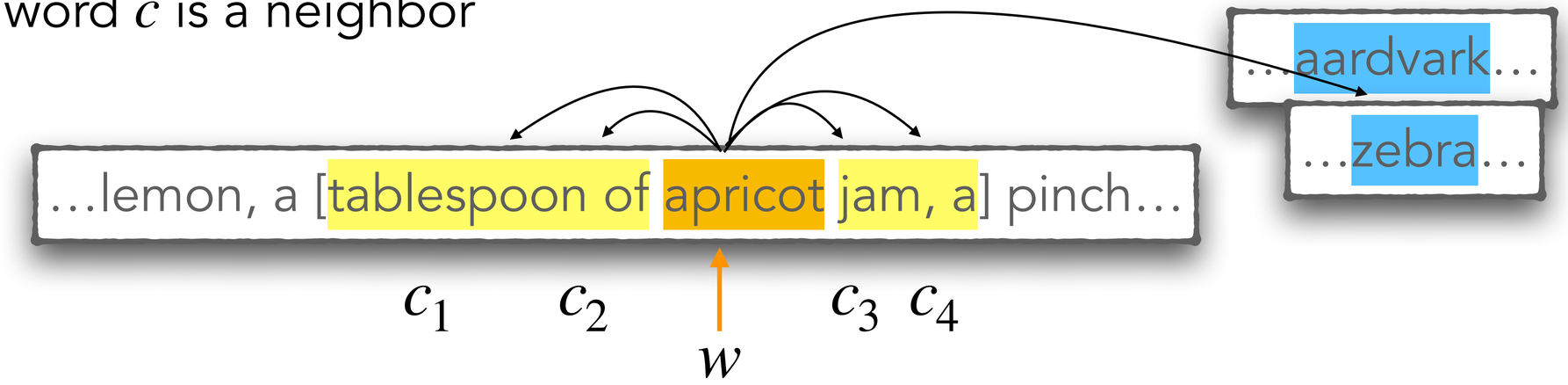
$$P(+|w,c)$$
$$P(-|w,c) = 1 - P(+|w,c)$$

word2vec: Pseudocode



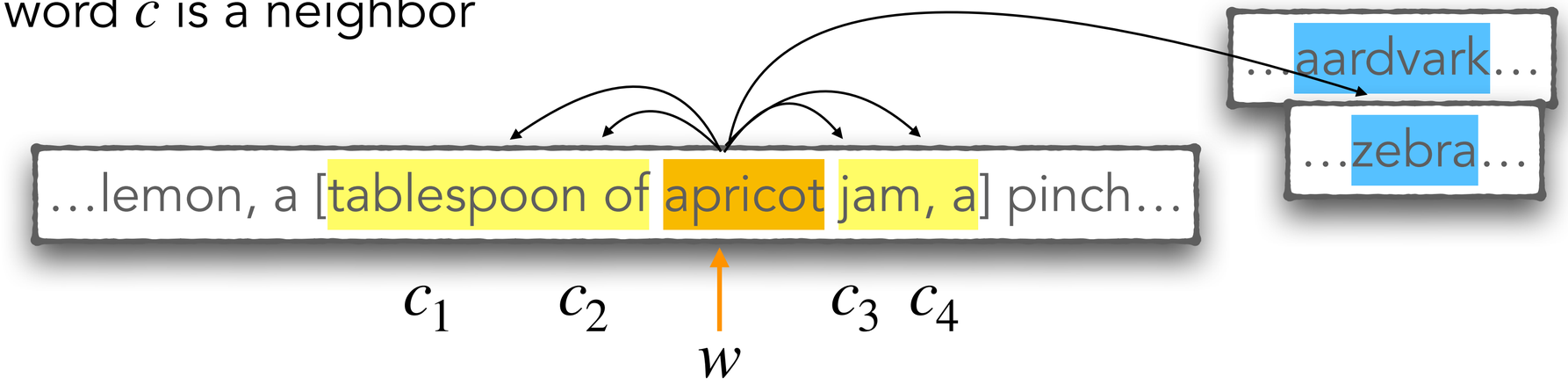
word2vec: Pseudocode

Predict if candidate word c is a neighbor



word2vec: Pseudocode

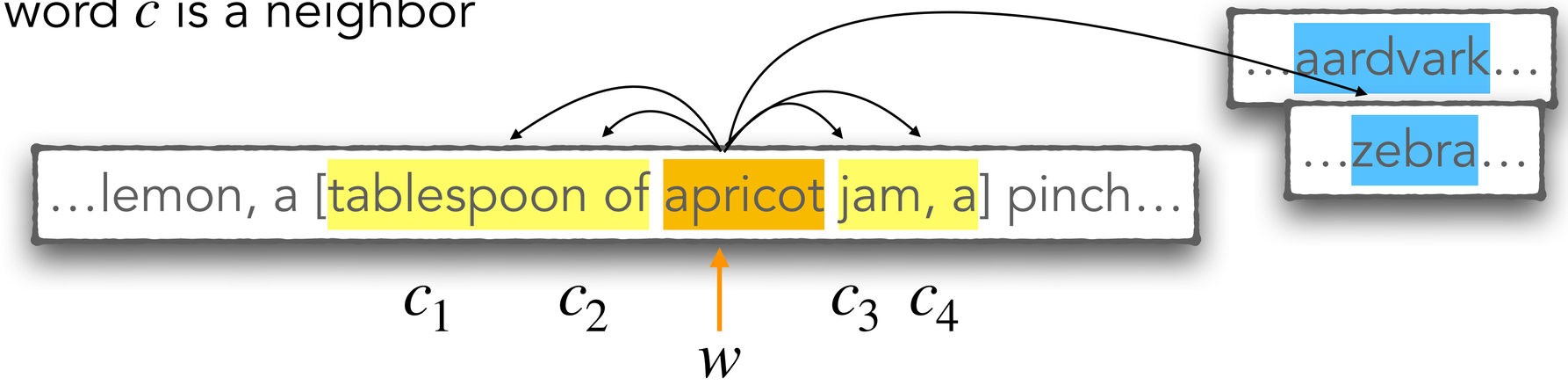
Predict if candidate word c is a neighbor



1. Treat the target word w and a neighboring context word c as **positive examples**.

word2vec: Pseudocode

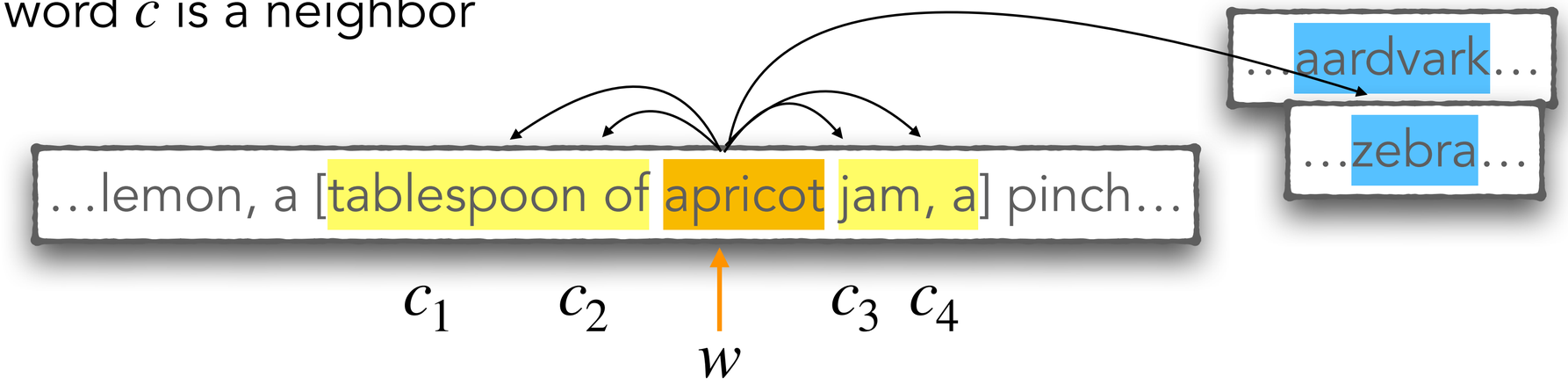
Predict if candidate word c is a neighbor



1. Treat the target word w and a neighboring context word c as **positive examples**.
2. Randomly sample other words in the lexicon to get **negative examples**

word2vec: Pseudocode

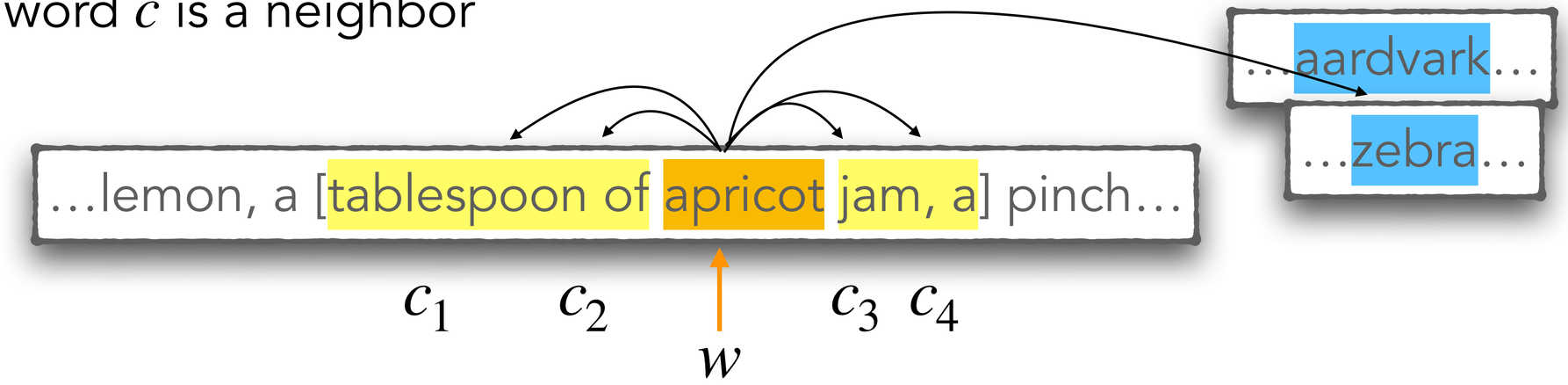
Predict if candidate word c is a neighbor



1. Treat the target word w and a neighboring context word c as **positive examples**.
2. Randomly sample other words in the lexicon to get **negative examples**
3. Use logistic regression to train a classifier to distinguish those two cases

word2vec: Pseudocode

Predict if candidate word c is a neighbor



1. Treat the target word w and a neighboring context word c as **positive examples**.
2. Randomly sample other words in the lexicon to get **negative examples**
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

word2vec: Probability Estimates

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

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product
    - Cosine similarity is just a normalized dot product

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product
  - Cosine similarity is just a normalized dot product
- So:

Can we just use cosine?

$$\text{sim}(w,c) \propto \mathbf{w} \cdot \mathbf{c}$$

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product
  - Cosine similarity is just a normalized dot product
- So:

Can we just use cosine?

$$\text{sim}(w,c) \propto \mathbf{w} \cdot \mathbf{c}$$

Vectors, not scalars!

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product
  - Cosine similarity is just a normalized dot product
- So:

Can we just use cosine?

- Still not a probability!

$$\text{sim}(w,c) \propto \mathbf{w} \cdot \mathbf{c}$$

Vectors, not scalars!

word2vec: Probability Estimates

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

- Central intuition: Base this probability on embedding similarity!
- Remember: two vectors are similar if they have a high dot product
  - Cosine similarity is just a normalized dot product
- So:

Can we just use cosine?

- Still not a probability!
  - We'll need to normalize to get a probability

$$\text{sim}(w,c) \propto \mathbf{w} \cdot \mathbf{c}$$

Vectors, not scalars!

Turning dot products into probabilities

Similarity:

$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

Turn into a probability using the sigmoid function:

Turning dot products into probabilities

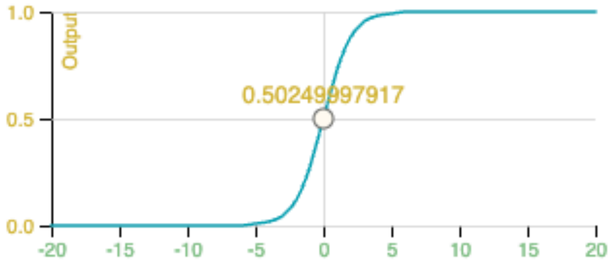
Similarity:

$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

Turn into a probability using the sigmoid function:

Sigmoid

$$f(0.01) = \frac{1}{1 + e^{-(0.01)}} = 0.50249997917$$



Turning dot products into probabilities

Similarity:

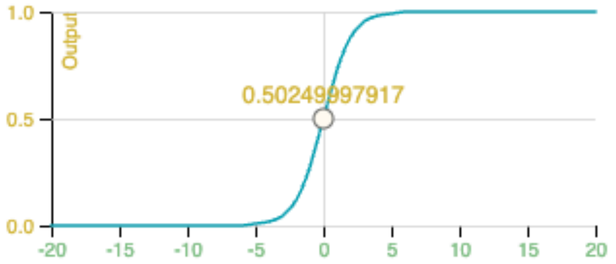
$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

Turn into a probability using the sigmoid function:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

Sigmoid

$$f(0.01) = \frac{1}{1 + e^{-(0.01)}} = 0.50249997917$$



Turning dot products into probabilities

Similarity:

$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

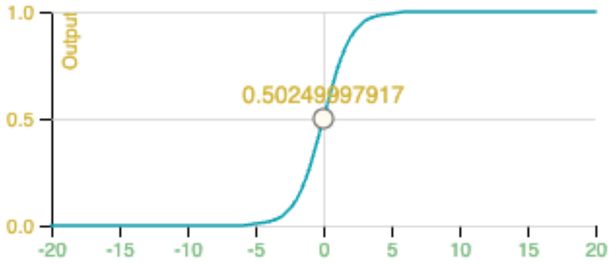
Turn into a probability using the sigmoid function:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

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

Sigmoid

$$f(0.01) = \frac{1}{1 + e^{-(0.01)}} = 0.50249997917$$



Turning dot products into probabilities

Similarity:

$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

Sigmoid

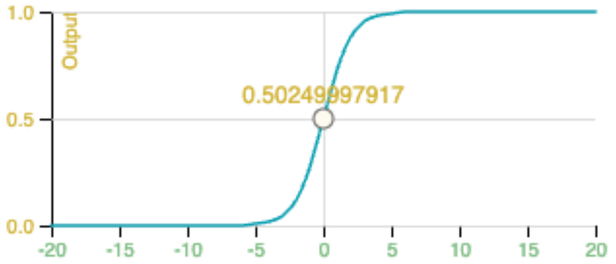
$$f(0.01) = \frac{1}{1 + e^{-(0.01)}} = 0.50249997917$$

Turn into a probability using the sigmoid function:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

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

$$= \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{c} \cdot \mathbf{w})}$$



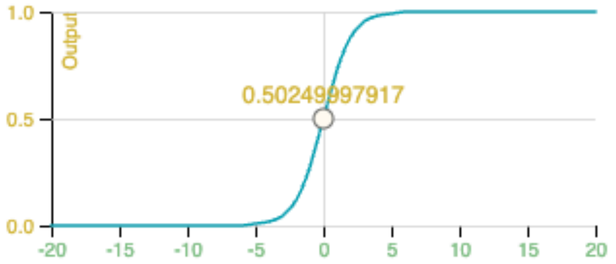
Turning dot products into probabilities

Similarity:

$$\text{sim}(w, c) \approx \mathbf{w} \cdot \mathbf{c}$$

Sigmoid

$$f(0.01) = \frac{1}{1 + e^{-(0.01)}} = 0.50249997917$$



Turn into a probability using the sigmoid function:

$$P(+|w, c) = \sigma(\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{c} \cdot \mathbf{w})}$$

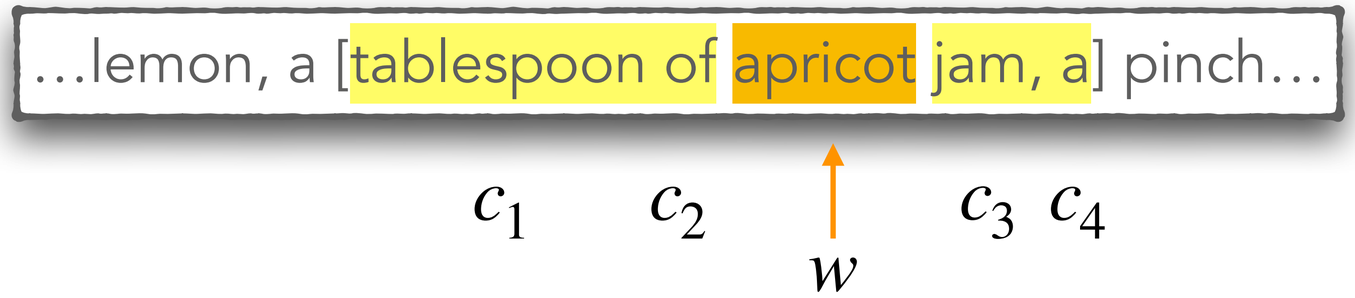
$$P(-|w, c) = 1 - P(+|w, c)$$

$$= \sigma(-\mathbf{c} \cdot \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{c} \cdot \mathbf{w})}$$

Logistic Regression!

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$



Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

But we have lots of context words

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

But we have lots of context words

- Depends on window size, L

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

But we have lots of context words

- Depends on window size, L
- We'll assume independence and just multiply them

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c}_i\cdot\mathbf{w})$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(\mathbf{c}_i\cdot\mathbf{w})$$

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

But we have lots of context words

- Depends on window size, L
- We'll assume independence and just multiply them

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L}\sigma(\mathbf{c}_i\cdot\mathbf{w})$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L}\log\sigma(\mathbf{c}_i\cdot\mathbf{w})$$

c_{neg} { …aardvark… …zebra…

Accounting for a context window

$$P(+|w,c) = \sigma(\mathbf{c}\cdot\mathbf{w}) = \frac{1}{1+\exp(-\mathbf{c}\cdot\mathbf{w})}$$

Single Context Word

…lemon, a [tablespoon of apricot jam, a] pinch…

c_1 c_2 w c_3 c_4

But we have lots of context words

- Depends on window size, L
- We'll assume independence and just multiply them

Same with negative context words!

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(\mathbf{c}_i\cdot\mathbf{w})$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log\sigma(\mathbf{c}_i\cdot\mathbf{w})$$

c_{neg} { …aardvark… …zebra…

$$\log P(-|w,c_{neg}) = \sum_{c'\in c_{neg}} \log\sigma(-\mathbf{c}'\cdot\mathbf{w})$$

word2vec classifier: Summary

word2vec classifier: Summary

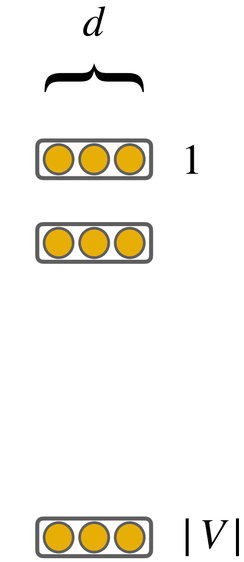
- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$

word2vec classifier: Summary

- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings)

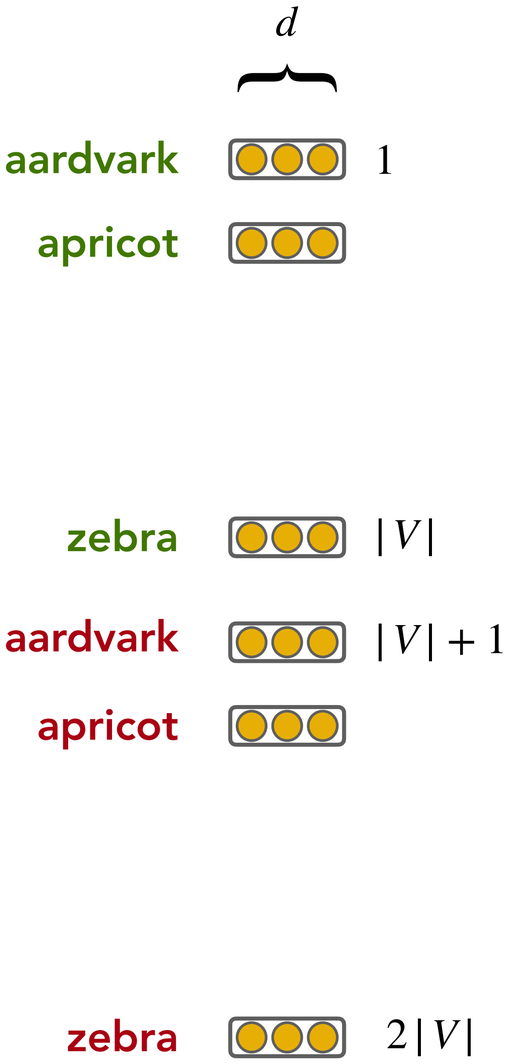
word2vec classifier: Summary

- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings)
- To compute this, we just need embeddings for all the words



word2vec classifier: Summary

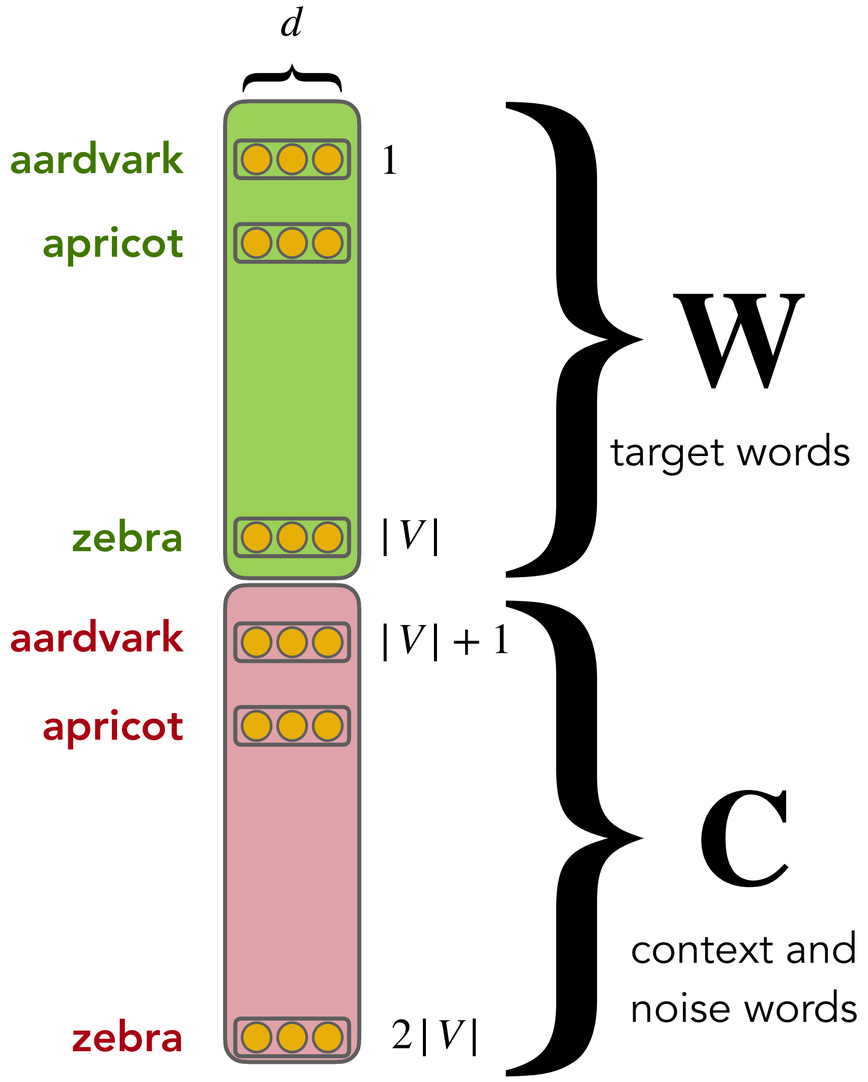
- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings)
- To compute this, we just need embeddings for all the words
  - Separate representations for targets and contexts



word2vec classifier: Summary

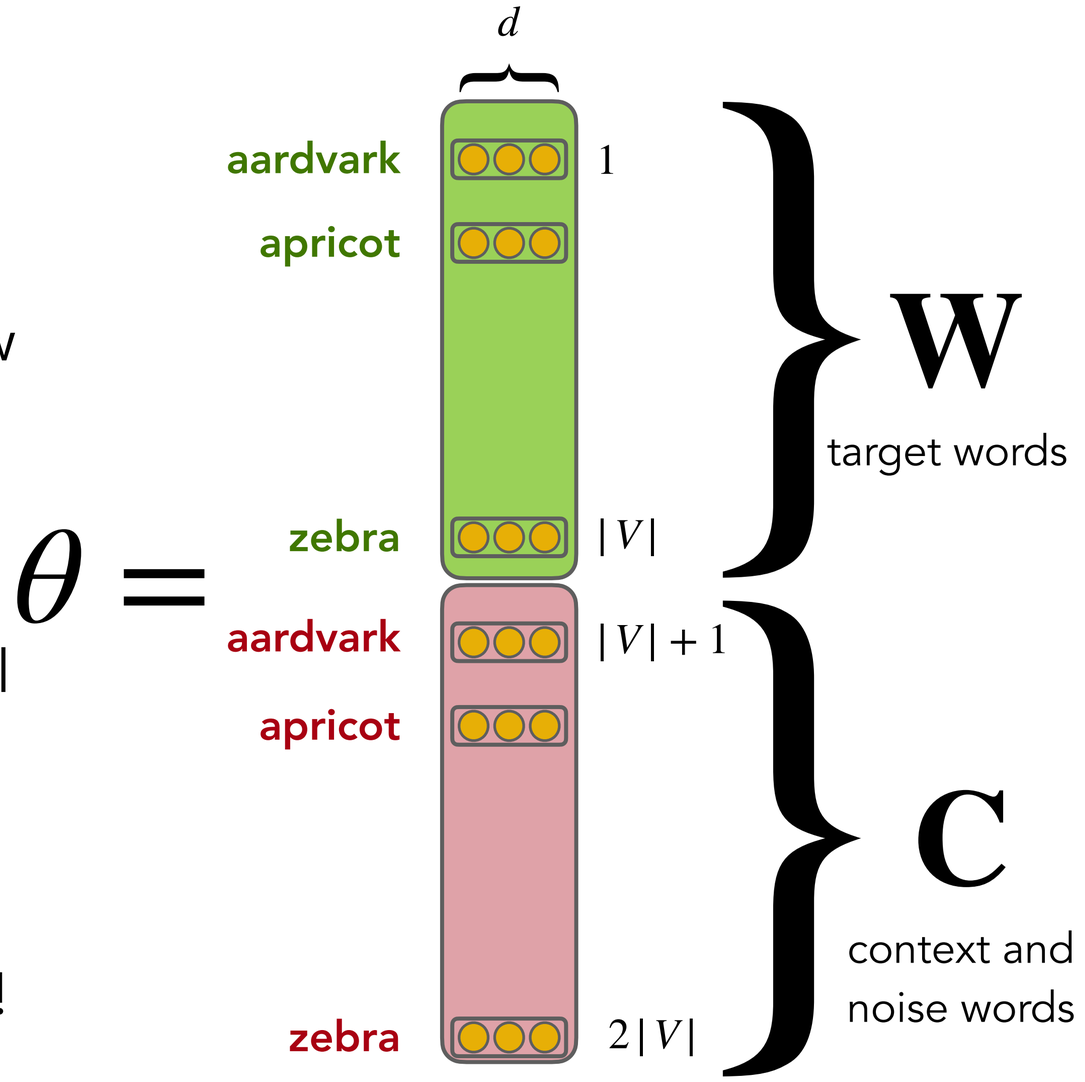
- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings)

- To compute this, we just need embeddings for all the words
  - Separate representations for targets and contexts



word2vec classifier: Summary

- A probabilistic classifier, given
  - a test target word w
  - its context window of L words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings)
- To compute this, we just need embeddings for all the words
  - Separate representations for targets and contexts
  - Same as the parameters we need to estimate!



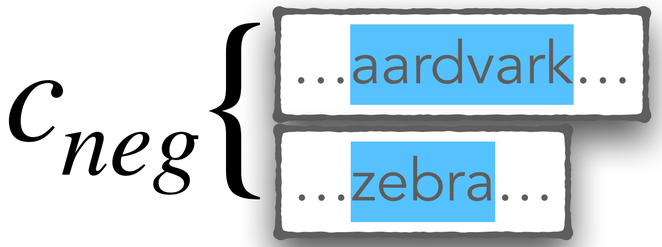
Lecture Outline

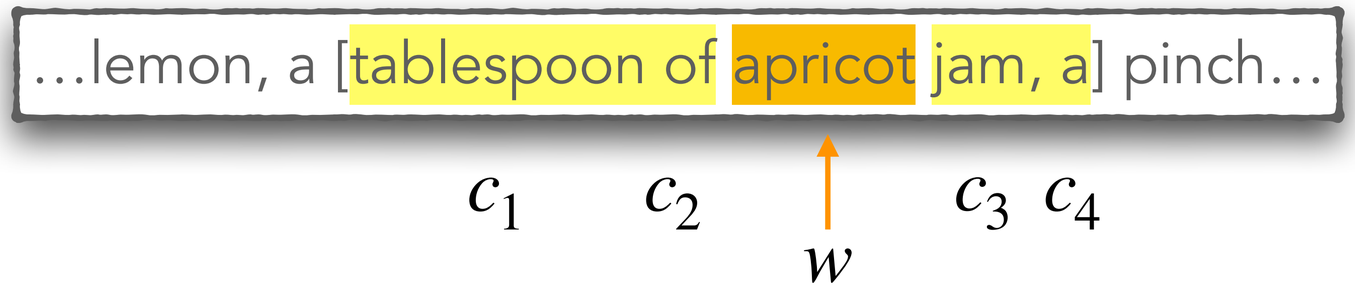
- Announcements
- Recap: Multinomial LR
- Recap: Lexical Semantics
- word2vec
  - Classification
  - Learning
- GloVe
- Properties and Evaluation of Word Embeddings

Learning word2vec embeddings

Word2vec: Training Data

For each positive example we'll grab a set of negative examples, sampling by weighted unigram frequency





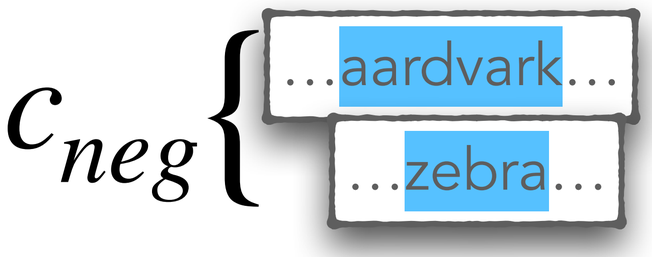
Negative examples

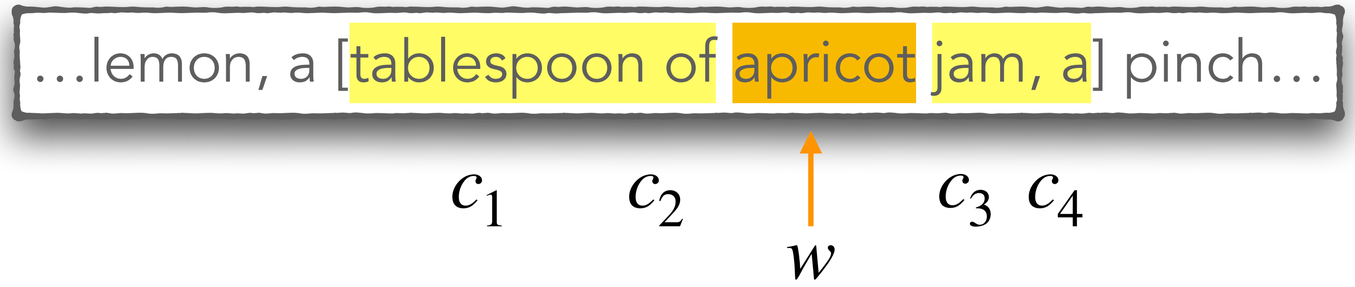
w	c_{neg}
apricot	aardvark
apricot	zebra
apricot	where
apricot	adversarial

Positive examples

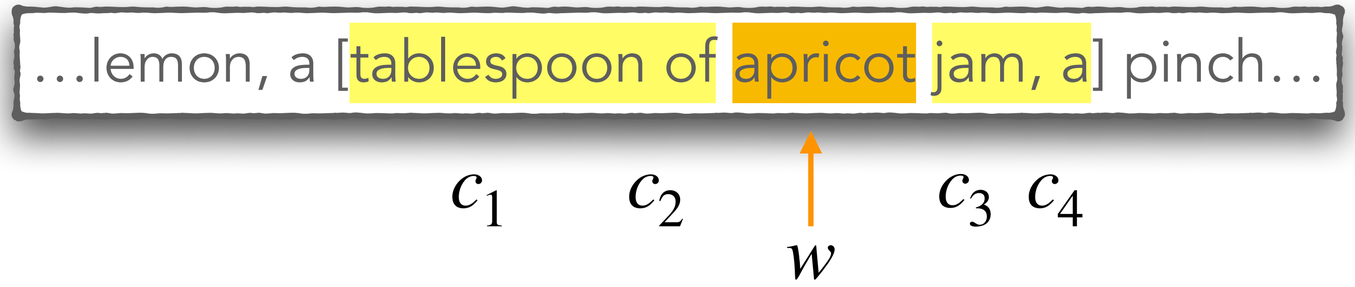
w	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

Word2vec: Learning Problem

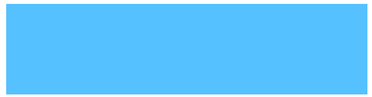




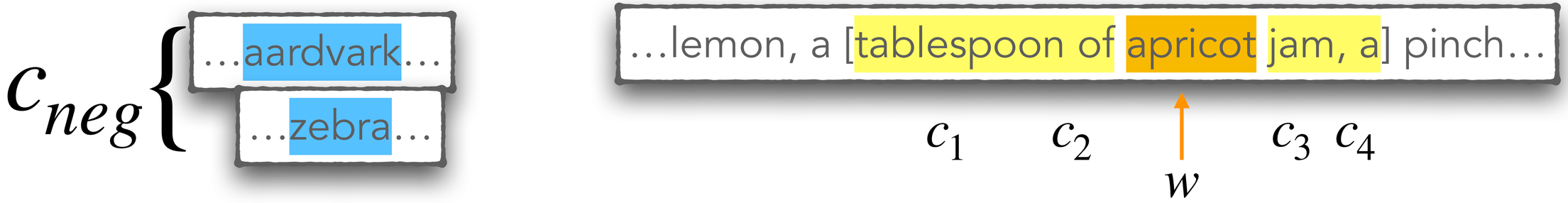
Word2vec: Learning Problem



Given



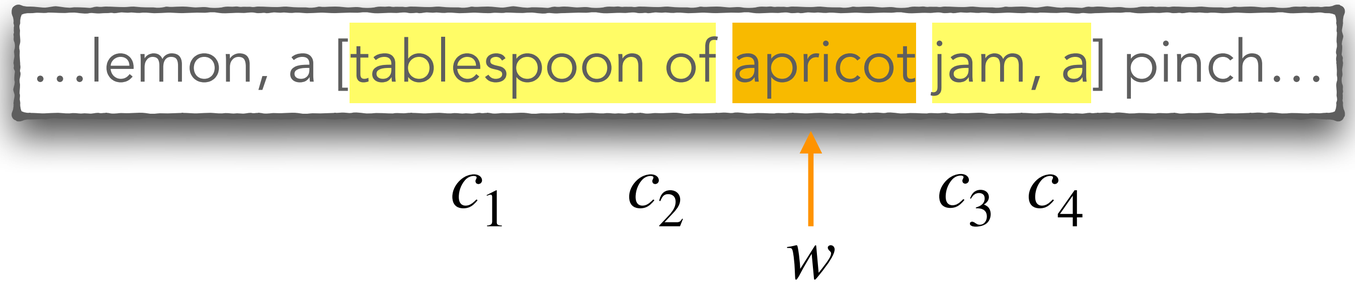
Word2vec: Learning Problem



Given

- the set of positive and negative training instances, and

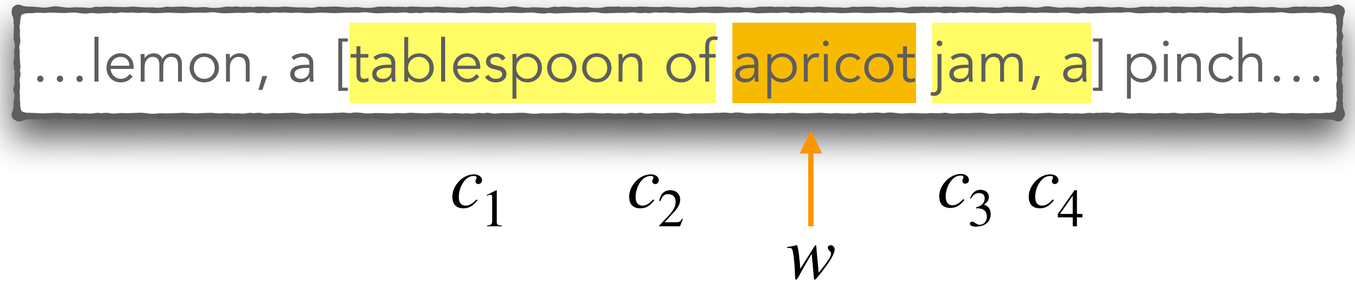
Word2vec: Learning Problem



Given

- the set of positive and negative training instances, and
- a set of randomly initialized embedding vectors of size $2|V|$,

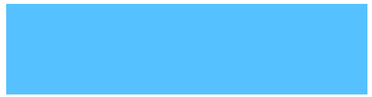
Word2vec: Learning Problem



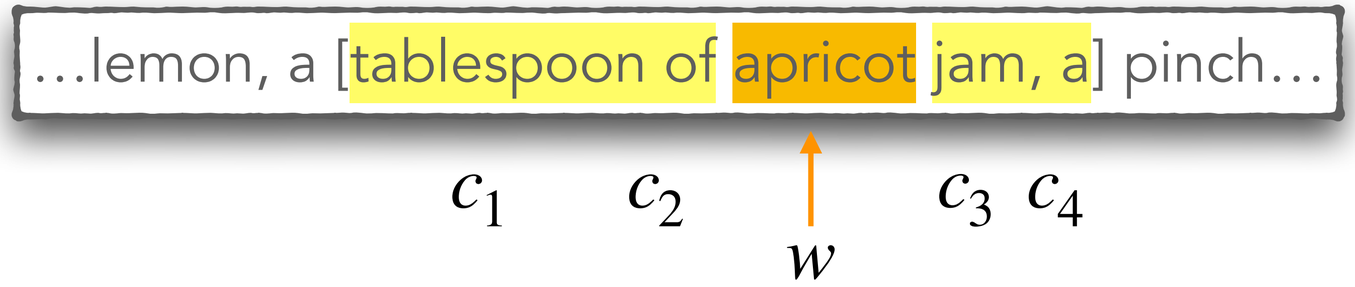
Given

- the set of positive and negative training instances, and
- a set of randomly initialized embedding vectors of size $2|V|$,

the goal of learning is to adjust those word vectors such that we:



Word2vec: Learning Problem



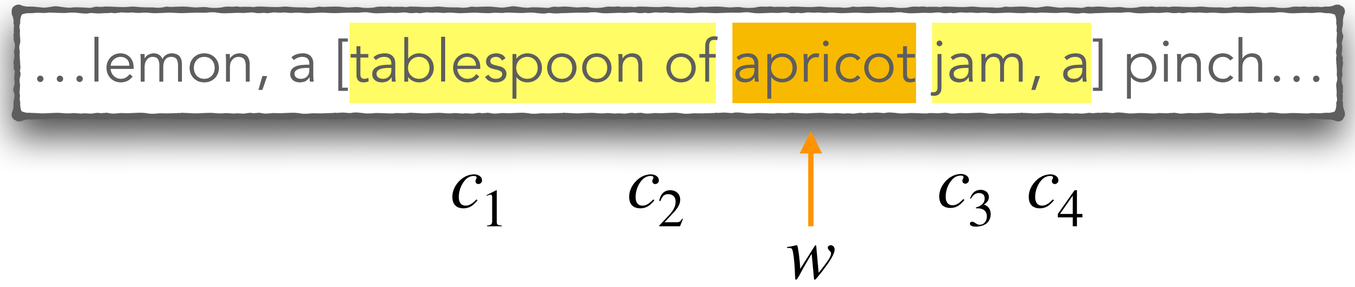
Given

- the set of positive and negative training instances, and
- a set of randomly initialized embedding vectors of size $2|V|$,

the goal of learning is to adjust those word vectors such that we:

- **Maximize** the similarity of the target word, context word pairs $(w, c_{1:L})$ drawn from the positive data

Word2vec: Learning Problem



Given

- the set of positive and negative training instances, and
- a set of randomly initialized embedding vectors of size $2|V|$,

the goal of learning is to adjust those word vectors such that we:

- **Maximize** the similarity of the target word, context word pairs $(w, c_{1:L})$ drawn from the positive data
- **Minimize** the similarity of the (w, c_{neg}) pairs drawn from the negative data

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word,
context pair…

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word, context pair…

$$L_{CE} = -\log[P(+|\mathbf{w}, \mathbf{c}_{pos})P(-|\mathbf{w}, \mathbf{c}_{neg})]$$

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word, context pair…

$$L_{CE} = -\log[P(+|\mathbf{w}, \mathbf{c}_{pos})P(-|\mathbf{w}, \mathbf{c}_{neg})]$$

Cross Entropy

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word, context pair…

$$L_{CE} = -\log[P(+|\mathbf{w}, \mathbf{c}_{pos})P(-|\mathbf{w}, \mathbf{c}_{neg})]$$

$$= -\left[\log P(+|\mathbf{w}, \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log P(-|\mathbf{w}, \mathbf{c}_{neg_j})\right]$$

Cross Entropy

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word, context pair…

$$L_{CE} = -\log[P(+|\mathbf{w}, \mathbf{c}_{pos})P(-|\mathbf{w}, \mathbf{c}_{neg})]$$

$$= -\left[\log P(+|\mathbf{w}, \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log P(-|\mathbf{w}, \mathbf{c}_{neg_j})\right]$$

$$= -\left[\log P(+|\mathbf{w}, \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log(1 - P(+|\mathbf{w}, \mathbf{c}_{neg_j})\right]$$

Cross Entropy

Loss function

Maximize the similarity of the target with the actual context words in a window of size L, and minimize the similarity of the target with the $K > L$ negative sampled non-neighbor words

For every word, context pair…

Cross Entropy

$$L_{CE} = -\log[P(+|\mathbf{w}, \mathbf{c}_{pos})P(-|\mathbf{w}, \mathbf{c}_{neg})]$$

$$= -\left[\log P(+|\mathbf{w}, \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log P(-|\mathbf{w}, \mathbf{c}_{neg_j})\right]$$

$$= -\left[\log P(+|\mathbf{w}, \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log(1 - P(+|\mathbf{w}, \mathbf{c}_{neg_j})\right]$$

$$= -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

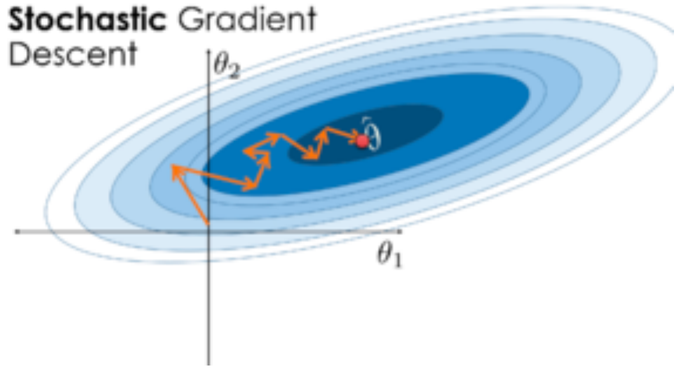
Learning the classifier

Learning the classifier

- How to learn?

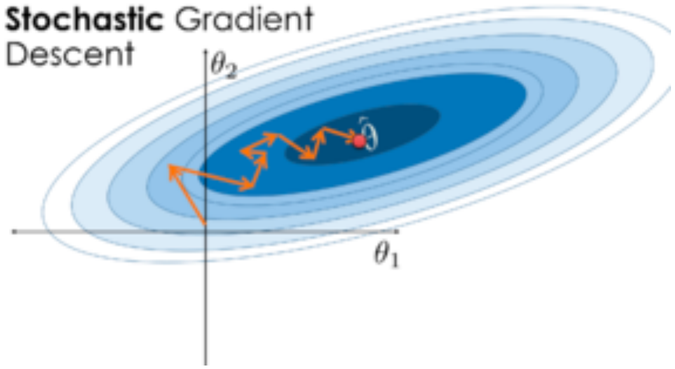
Learning the classifier

- How to learn?
  - Stochastic gradient descent!



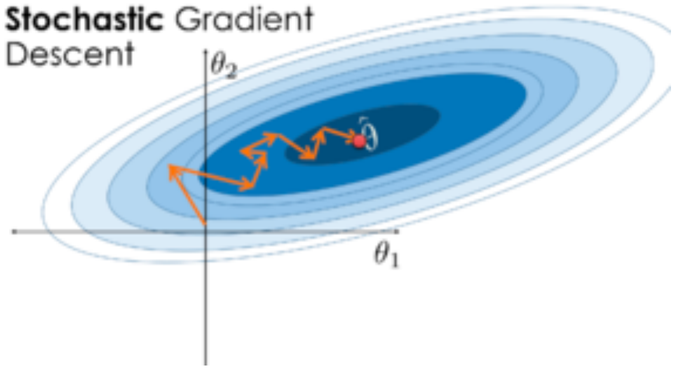
Learning the classifier

- How to learn?
  - Stochastic gradient descent!
  - Iterative process



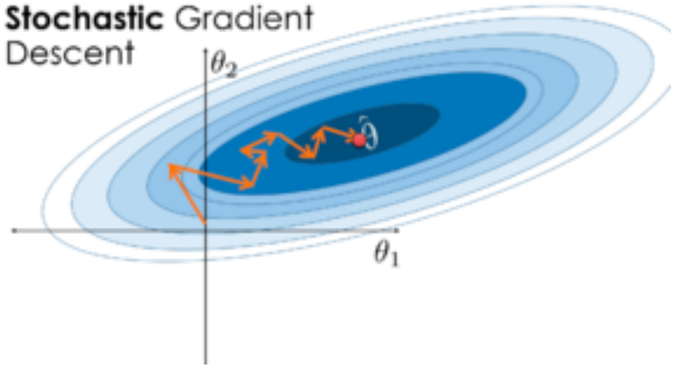
Learning the classifier

- How to learn?
  - Stochastic gradient descent!
  - Iterative process
  - Start with randomly initialized weights
    - Update the parameters by computing gradients of the loss w.r..t. parameters

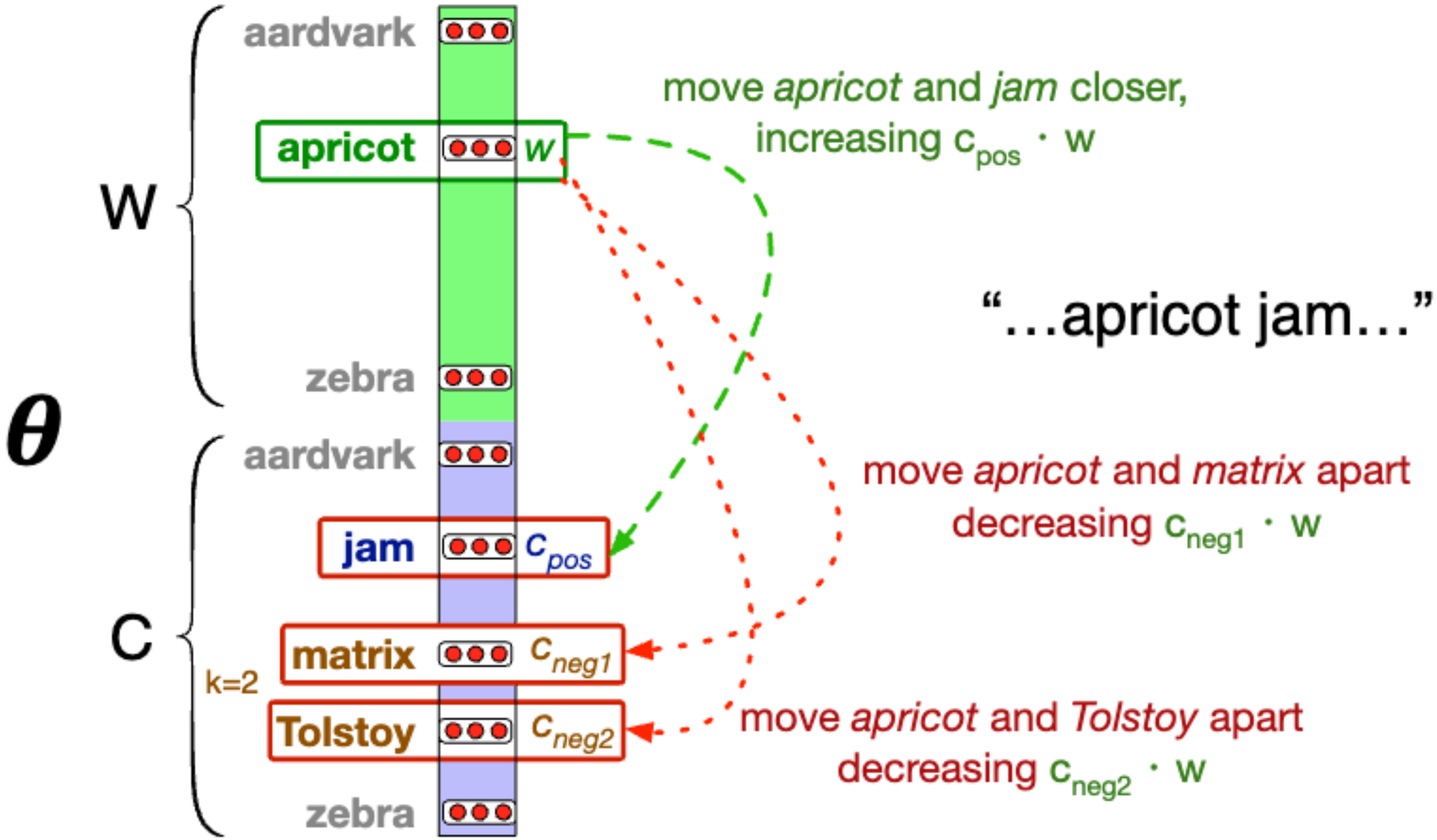


Learning the classifier

- How to learn?
  - Stochastic gradient descent!
  - Iterative process
  - Start with randomly initialized weights
    - Update the parameters by computing gradients of the loss w.r..t. parameters
    - Stop when the parameters (or, the loss) do not change much…
- We'll adjust the word weights to
  - make the positive pairs more likely
  - and the negative pairs less likely,
  - over the entire training set.



Intuition of one step of gradient descent



SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

3 different parameters

SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

3 different parameters

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{w}$$

SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

3 different parameters

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{neg_j}} = [\sigma(\mathbf{c}_{neg_j} \cdot \mathbf{w})]\mathbf{w}$$

SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

3 different parameters

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{neg_j}} = [\sigma(\mathbf{c}_{neg_j} \cdot \mathbf{w})]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{c}_{pos} + \sum_{j=1}^{K} [\sigma(\mathbf{c}_{neg_j} \cdot \mathbf{w})]\mathbf{c}_{neg_j}$$

SGD: Derivates

$$L_{CE} = -\left[\log \sigma(\mathbf{w} \cdot \mathbf{c}_{pos}) + \sum_{j=1}^{K} \log \sigma(-\mathbf{w} \cdot \mathbf{c}_{neg_j})\right]$$

3 different parameters

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial \mathbf{c}_{neg_j}} = [\sigma(\mathbf{c}_{neg_j} \cdot \mathbf{w})]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{c}_{pos} + \sum_{j=1}^{K} [\sigma(\mathbf{c}_{neg_j} \cdot \mathbf{w})]\mathbf{c}_{neg_j}$$

Update the parameters by subtracting respective η-weighted gradients