Vector Semantics and Embeddings

Word Meaning

What do words mean?

Stanford

What do words mean?

N-gram or text classification methods we've seen so far

- Words are just strings (or indices w_i in a vocabulary list)
- That's not very satisfactory!

Stanford

What do words mean?

N-gram or text classification methods we've seen so far
- Words are just strings (or indices w_i in a vocabulary list)
- That's not very satisfactory!

Introductory logic classes:
- The meaning of "dog" is DOG; cat is CAT

$$\forall x \, \mathrm{DOG}(x) \longrightarrow \mathrm{MAMMAL}(x)$$

Stanford

Desiderata

What should a theory of word meaning do for us?

Let's look at some desiderata

From lexical semantics, the linguistic study of word meaning

Stanford

Lemmas and senses

mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Modified from the online thesaurus WordNet

Stanford

Lemmas and senses

lemma

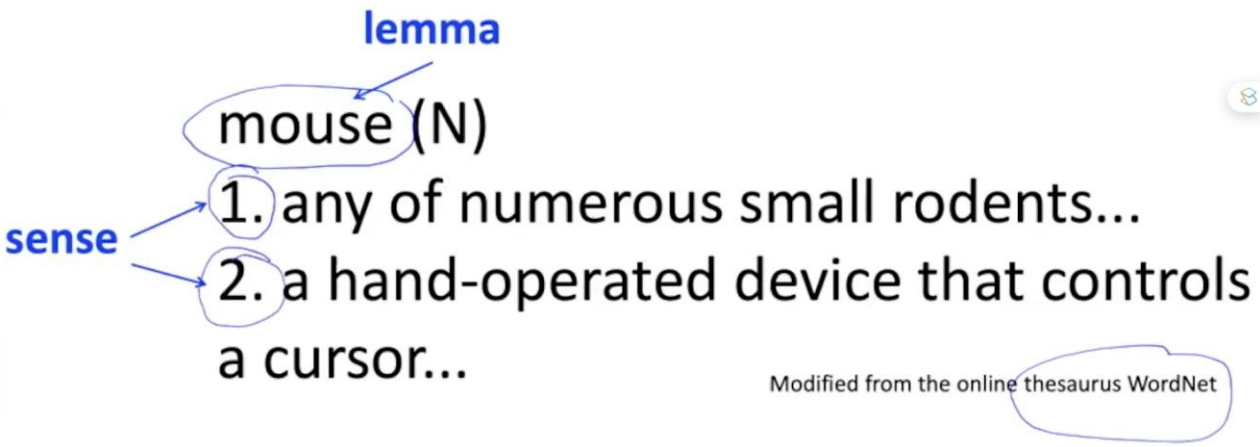
mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Modified from the online thesaurus WordNet

Stanford

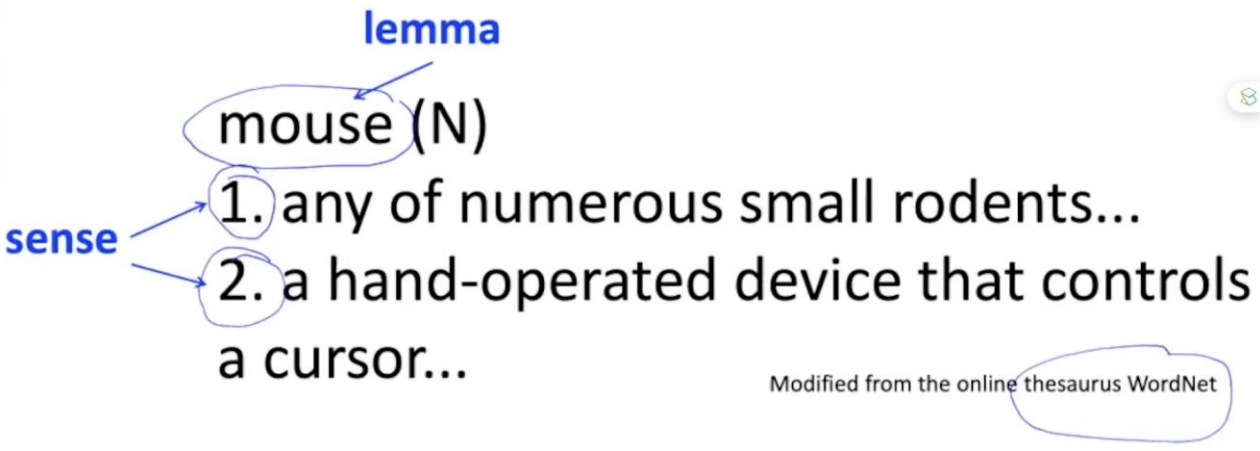
Lemmas and senses



Modified from the online thesaurus WordNet

Stanford

Lemmas and senses



A sense or "concept" is the meaning component of a word
Lemmas can be polysemous (have multiple senses)

Relations between senses: Synonymy

Synonyms have the same meaning in some or all contexts.

- filbert / hazelnut
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- water / H_20

Stanford

Relations between senses: Synonymy

Note that there are probably no examples of perfect synonymy.

- Even if many aspects of meaning are identical
- Still may differ based on politeness, slang, register, genre, etc.

Stanford

Relation: Synonymy?

water/H_20

Stanford

Relation: Synonymy?

water/H_20

"H_20" in a surfing guide?

big/large

my big sister != my large sister

Stanford

The Linguistic Principle of Contrast

Difference in form → difference in meaning

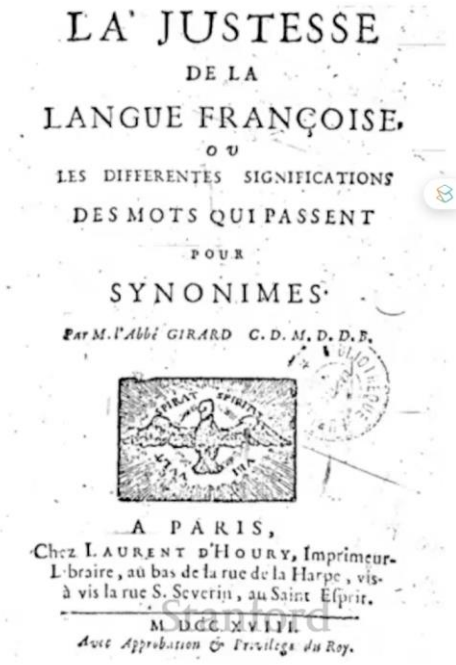
Stanford

Abbé Gabriel Girard 1718

Re: "exact" synonyms

"je ne crois pas qu'il y ait de mot ſynonime dans aucune Langue."

[I do not believe that there is a synonymous word in any language]



LA JUSTESSE
DE LA
LANGUE FRANÇOISE,
OU
LES DIFFERENTES SIGNIFICATIONS
DES MOTS QUI PASSENT
POUR
SYNONIMES.

Par M. l'Abbé GIRARD C. D. M. D. D. F.

A PARIS,
Chez LAURENT D'HOURY, Imprimeur-Libraire, au bas de la rue de la Harpe, vis-à-vis la rue S. Severin, au Saint Eſprit.

M. DCC. XVIII.
Avec Approbation & Privilege du Roy.

Thanks to Mark Aronoff!

Relation: Similarity

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

`car, bicycle`

`cow, horse`

Stanford

Ask humans how similar 2 words are

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

Stanford

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

Relation: Word relatedness

Also called "word association"

Words can be related in any way, perhaps via a semantic frame or field

- `coffee, tea:` **similar**
- `coffee, cup:` **related**, not similar

Stanford

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

Stanford

Relation: Antonymy

Senses that are opposites with respect to only one feature of meaning

Otherwise, they are very similar!

Stanford

Relation: Antonymy

Senses that are opposites with respect to only one feature of meaning

Otherwise, they are very similar!

dark/light short/long fast/slow rise/fall
hot/cold up/down in/out

Stanford

Relation: Antonymy

Senses that are opposites with respect to only 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`

Stanford

Connotation (sentiment)

- Words have **affective** meanings
  - Positive connotations (*happy*)
  - Negative connotations (*sad*)
- Connotations can be subtle:
  - Positive connotation: *copy, replica, reproduction*
  - Negative connotation: *fake, knockoff, forgery*
- Evaluation (sentiment!)
  - Positive evaluation (*great*, *love*)
  - Negative evaluation (*terrible*, *hate*)

Stanford

Connotation

Osgood et al. (1957)

Words seem to vary along 3 affective dimensions:

- **valence**: the pleasantness of the stimulus
- **arousal**: the intensity of emotion provoked by the stimulus
- **dominance**: the degree of control exerted by the stimulus

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Values from NRC VAD Lexicon (Mohammad 2018)

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
- Connotation

Stanford

Vector Semantics

Computational models of word meaning

Can we build a theory of how to represent word meaning, that accounts for at least some of the desiderata?

We'll introduce **vector semantics**

- The standard model in language processing!
- Handles many of our goals!

Stanford

Ludwig Wittgenstein

PI #43:

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

Stanford

Let's define words by their usages

One way to define "usage":

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.

Stanford

What does recent English borrowing *ongchoi* mean?

Stanford

What does recent English borrowing *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

Stanford

What does recent English borrowing *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

Stanford

What does recent English borrowing *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
  - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Ongchoi: *Ipomoea aquatica "Water Spinach"*

空心菜
kangkong
rau muống
…

Yamaguchi, Wikimedia Commons, public domain

Idea 1: Defining meaning by linguistic distribution

Let's define the meaning of a word by its distribution in language use, meaning its neighboring words or grammatical environments.

Stanford

Idea 2: Meaning as a point in space (Osgood et al. 1957)

3 affective dimensions for a word

- **valence**: pleasantness
- **arousal**: intensity of emotion
- **dominance**: the degree of control exerted

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

NRC VAD Lexicon
(Mohammad 2018)

Stanford

Idea 2: Meaning as a point in space (Osgood et al. 1957)

3 affective dimensions for a word

- **valence**: pleasantness
- **arousal**: intensity of emotion
- **dominance**: the degree of control exerted

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

NRC VAD Lexicon
(Mohammad 2018)

Stanford

Hence the connotation of a word is a vector in 3-space

Idea 1: Defining meaning by linguistic distribution

Idea 2: Meaning as a point in multidimensional space

Stanford

Defining meaning as a point in space based on distribution

Each word = a vector (not just "good" or "w_{45}")

Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**

Stanford

Defining meaning as a point in space based on distribution

Each word = a vector (not just "good" or "w_{45}")

Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**

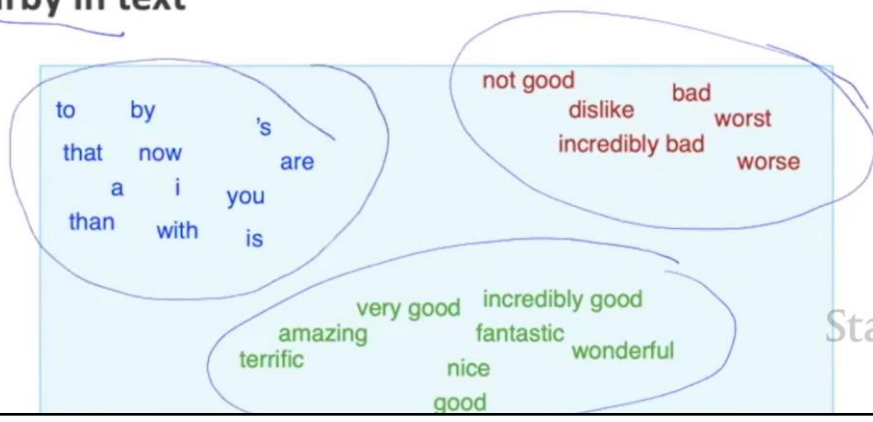
Stanford

Defining meaning as a point in space based on distribution

Each word = a vector (not just "good" or "w_{45}")

Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**



Stanford

We define meaning of a word as a vector

Called an "embedding" because it's embedded into a space (see textbook)

The standard way to represent meaning in NLP

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

Stanford

We define meaning of a word as a vector

Called an "embedding" because it's embedded into a space (see textbook)

The standard way to represent meaning in NLP

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

Fine-grained model of meaning for similarity

Stanford

Intuition: why vectors?

Consider sentiment analysis:

- With **words**, a feature is a word identity
  - Feature 5: 'The previous word was "terrible"'
  - requires **exact same word** to be in training and test

Stanford

Intuition: why vectors?

Consider sentiment analysis:

- With **words**, a feature is a word identity
  - Feature 5: 'The previous word was "terrible"'
  - requires **exact same word** to be in training and test
- With **embeddings**:
  - Feature is a word vector
  - 'The previous word was vector [35,22,17…]
  - Now in the test set we might see a similar vector [34,21,14]
  - We can generalize to **similar but unseen** words!!!

Stanford

Intuition: why vectors?

Consider sentiment analysis:

- With **words**, a feature is a word identity
  - Feature 5: 'The previous word was "terrible"'
  - requires **exact same word** to be in training and test
- With **embeddings**:
  - Feature is a word vector
  - 'The previous word was vector [35,22,17…]
  - Now in the test set we might see a similar vector [34,21,14]
  - We can generalize to **similar but unseen** words!!!

Stanford

We'll discuss 2 kinds of embeddings

tf-idf

- Information Retrieval workhorse!
- A common baseline model
- **Sparse** vectors
- Words are represented by (a simple function of) the **counts** of nearby words

Stanford

We'll discuss 2 kinds of embeddings

tf-idf

- Information Retrieval workhorse!
- A common baseline model
- **Sparse** vectors
- Words are represented by (a simple function of) the **counts** of nearby words

Word2vec

- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
- Later we'll discuss extensions called **contextual embeddings**

Stanford

From now on:
Computing with meaning representations instead of string representations

荃者所以在鱼，得鱼而忘荃 Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言 Words are for meaning;
Once you get the meaning, you can forget the words

庄子(Zhuangzi), Chapter 26

Stanford

Words and Vectors

Term-document matrix

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

Stanford

how to represent words as simple vectors of context counts. Imagine we have a collection of documents such as all the works of Shakespeare.
We can represent documents in such a collection by a term document matrix in which each row represents a word in the vocabulary and each column represents a document in the collection.

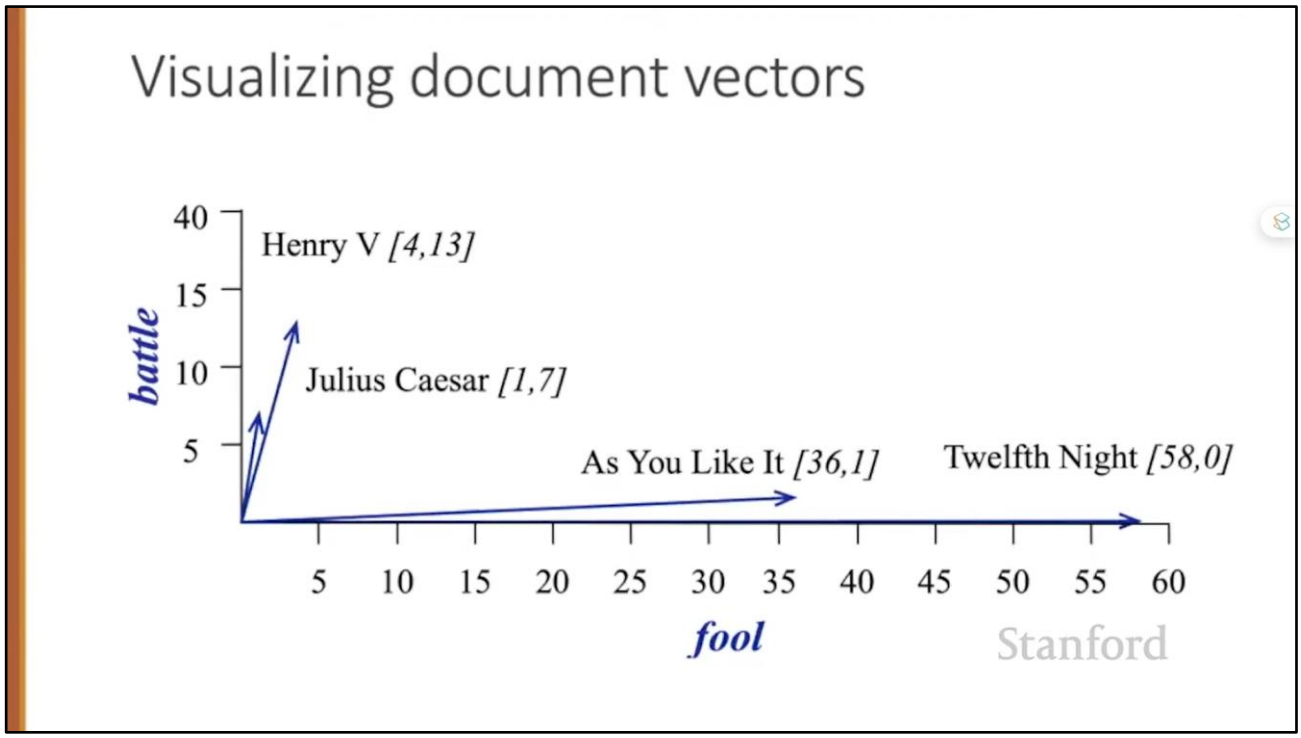
Term-document matrix

Each document is represented by a vector of words

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

Stanford

matrix showing the occurrence of four words and four plays by Shakespeare. Each cell in this matrix represents the number of times a particular word defined by the row occurs in a particular document defined by the column. Thus, Fool appears fifty eight times in Twelfth Night and we can think of each column as a vector, representing a document as a point in |V| dimensional space.



Here's a spatial visualization of the document vectors from those four Shakespeare play documents.
Here we're seeing just two of the dimensions corresponding to the words battle and fool.
And notice that the comedies have high values for the "fool" dimension and low values for the "battle" dimension.

Vectors are the basis of information retrieval

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

Vectors are similar for the two comedies

But comedies are different than the other two

Stanford

Two documents that are similar will tend to have similar words, and if two documents have similar words there, column vectors will tend to be similar.

Vectors are the basis of information retrieval

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

Vectors are similar for the two comedies

But comedies are different than the other two

Comedies have more *fools* and *wit* and fewer *battles*.

So the vectors for the comedies As You Like It and Twelfth Night look a lot more like each other.
More fools than wit and less battles than they look like Julius Caesar or Henry the Fifth.
A real term document matrix, wouldn't just have four rows and columns, let alone two more generally, the term document matrix has |V| rows, one for each word type in the vocabulary. And D columns, one for each document in the collection.

Idea for word meaning: Words can be vectors too!!!

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

Stanford

Now, here's a new idea, vector semantics can also be used to represent the meaning of words, not just documents.
And we do this by associating each word with a word vector. Now a row vector rather than a column vector.

Idea for word meaning: Words can be vectors too!!!

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

battle is "the kind of word that occurs in Julius Caesar and Henry V"

fool is "the kind of word that occurs in comedies, especially Twelfth Night"

Stanford

So the four dimensions of the vector for fool, for example. Thirty six fifty eight one four correspond to the four Shakespeare plays for documents.
We saw that similar documents had similar vectors because similar documents have similar words
And the same principle applies to words similar words have similar vectors because they tend to occur in similar documents.
The term document matrix thus lets us represent the meaning of a word by the document it tends to occur in.

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

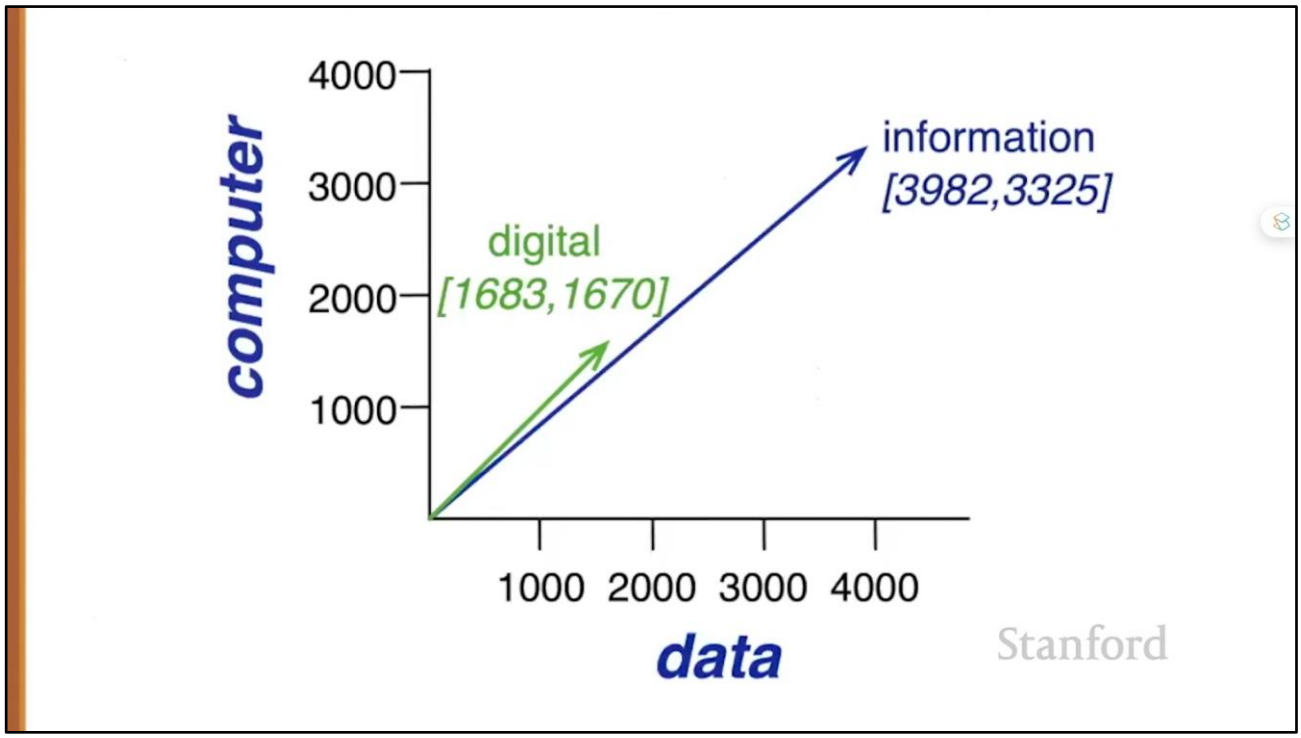
Two **words** are similar in meaning if their context vectors are similar

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

	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	...

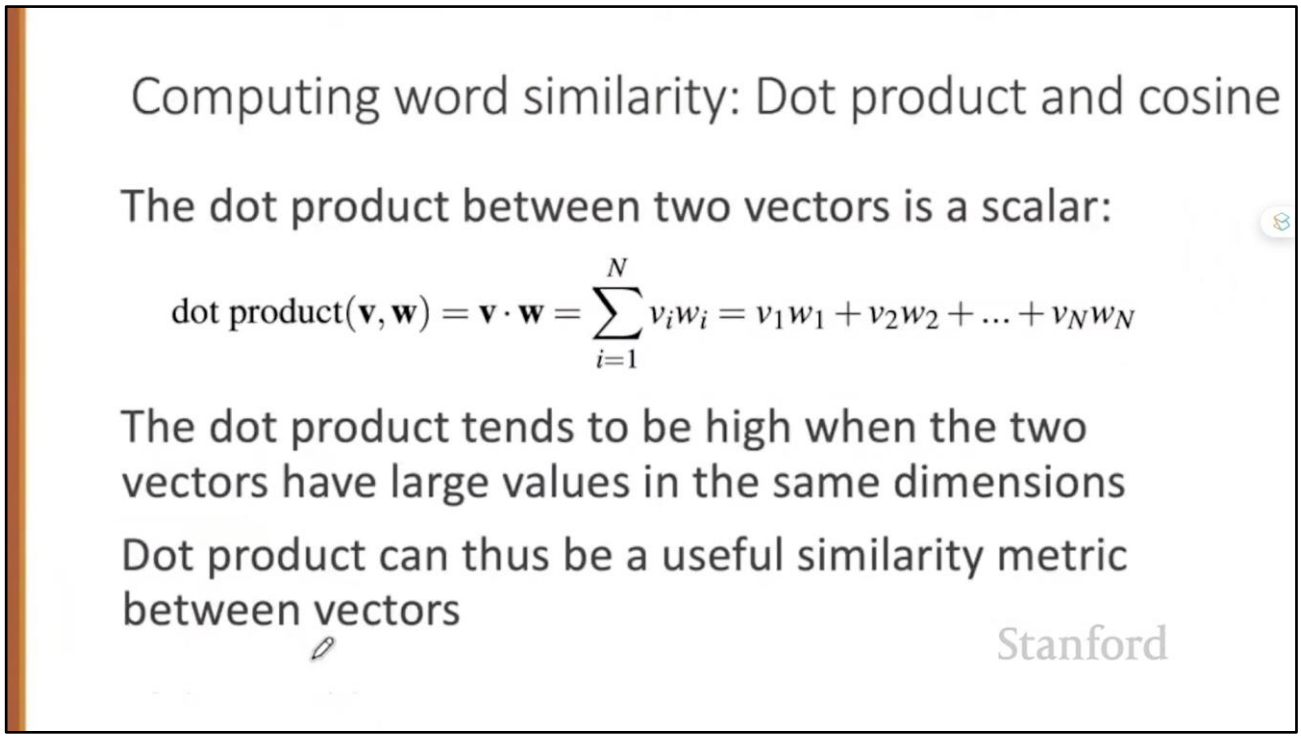
An alternative to using the term document matrix to represent words as vectors of document counts is to use the term matrix
also called the word word matrix or the term context matrix in which the columns are labeled by words rather than documents.
So the Matrix is now a V by V, and each cell records the number of times the target word in the row and the context word in the column co-occur in some context.
The context could be the document, in which case the cell, let's say, digital and computer, represents the number of times the two words appear in the same document.
It's more common, however, to use smaller contexts. Generally, a window around the word, for example, four words to the left and forwards to the right.
And in that case, the cell represents the number of times in some training corpus. The column word occurs in such a plus or minus forward window around the row word.
Here is one example. Each of four words in their windows. And then from examples like this, we compute this table showing how often each of these row words, how often they occur in a plus or minus forward window around each of these context words.
So, for example, the 1670 here means the word digital occurred within four words on either side of the word computer in this corpus 1670 times.

Note that with this definition of vectors, the vector for digital and the vector for information are more similar to each other.
High values for count and data and low values for pie and sugar.
Then either of them is say to strawberry, which is opposite appearing more in the context of pie in sugar.



Here's a spatial visualization of word vectors for digital and information. showing just two of the dimensions corresponding to the words data and computer.
Of course, in real life, these vectors aren't just of length to V. The length of the vector is generally the size of the vocabulary.
Often between 10000 and 50000 words, perhaps using the most frequent words in the training corpus, keeping words after about the most frequent fifty thousand or so is generally not helpful.
Since most of these numbers are zero, these are sparse vectors and there are efficient algorithms for storing and computing with sparse matrices.

Cosine Similarity



To measure similarity between two words, we need a metric that compares their representations.
By far, the most common similarity metric is the cosine of the angle between the two vectors.
The cosine, like most measures for vector similarity used in NLP, is based on the dot product operator from Linear Algebra, also called the inner product, in which we multiply the vectors element wise and add up to get a single scalar value.

Computing word similarity: Dot product and cosine

The dot product between two vectors is a scalar:

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

The dot product tends to be high when the two vectors have large values in the same dimensions

Dot product can thus be a useful similarity metric between vectors

Stanford

The dot-product acts as a similarity metric because it tend to be high just when the two vectors have large values in the same dimension.
Alternatively, vectors that have zeros in different dimension
orthogonal vectors will have a dot product of zero representing their strong dissimilarity.

Problem with raw dot-product

Dot product favors long vectors

Dot product is higher if a vector is longer (has higher values in many dimension)

Vector length:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Frequent words (of, the, you) have long vectors (since they occur many times with other words).

So dot product overly favors frequent words

Stanford

The raw dot product, however, has a problem as a similarity metric. It favors long vectors.

Problem with raw dot-product

Dot product favors long vectors

Dot product is higher if a vector is longer (has higher values in many dimension)

Vector length:

$$|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$$

Frequent words (of, the, you) have long vectors (since they occur many times with other words).

So dot product overly favors frequent words

Stanford

8

Alternative: cosine for computing word similarity

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

Based on the definition of the dot product between two vectors a and b

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$
$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

Stanford

So we modify the dot product to normalize for the vector length by dividing the dot product by the length of each of the two vectors.

Alternative: cosine for computing word similarity

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

Based on the definition of the dot product between two vectors a and b

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$
$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

Stanford

And this normalized dot product turns out to be the same as the cosine of the angle between the two vectors.

Alternative: cosine for computing word similarity

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

Based on the definition of the dot product between two vectors a and b

$$\mathbf{a}\cdot\mathbf{b} = |\mathbf{a}||\mathbf{b}|\cos\theta$$
$$\frac{\mathbf{a}\cdot\mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos\theta$$

Stanford

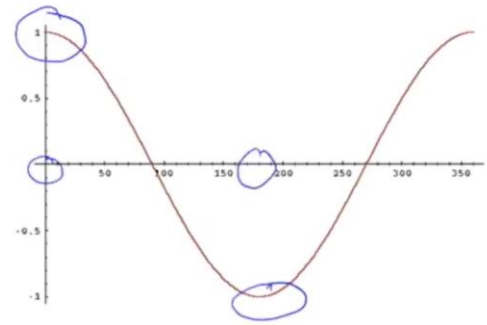
This is based on the geometric definition of the dot product as the product of Euclidean magnitudes of the two vectors and the coastline of the angle between them.

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



Stanford

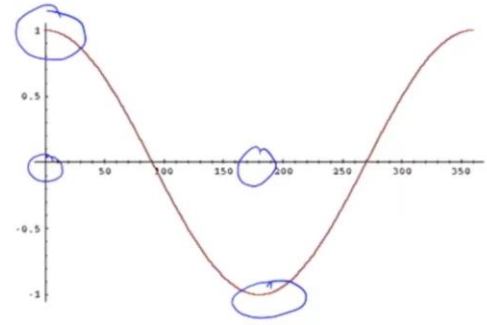
The cosine value ranges from one for vectors pointing in the same direction with an angle of zero between them to minus one for vectors pointing in opposite directions,

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



But since raw frequency values are non-negative, the cosine for term-term matrix vectors ranges from 0–1

Stanford

But since raw frequency values are non-negative, the cosine for these vectors ranges from 0 to 1.

Cosine examples

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

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

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

Let's see how the cosine computes, which of the words cherry or digital is closer in meaning to information.
Just using raw counts from the shortened table to compute the cosine between cherry and information.

Cosine examples

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

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

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

We take the dot product between these two vectors. So that will be 442 times 5. Plus 8 times 3982 plus 2 times 3325 in the numerator and then the denominator.

Cosine examples

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

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

$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

Stanford

We have the length of the two vectors. First, the length of cherry, which is the square root of the sum of 442 squared plus eight squared plus two squared. And then the length of information, the square root of five squared plus 3982. squared plus 3325 squared. And we get a result: .017

Cosine examples

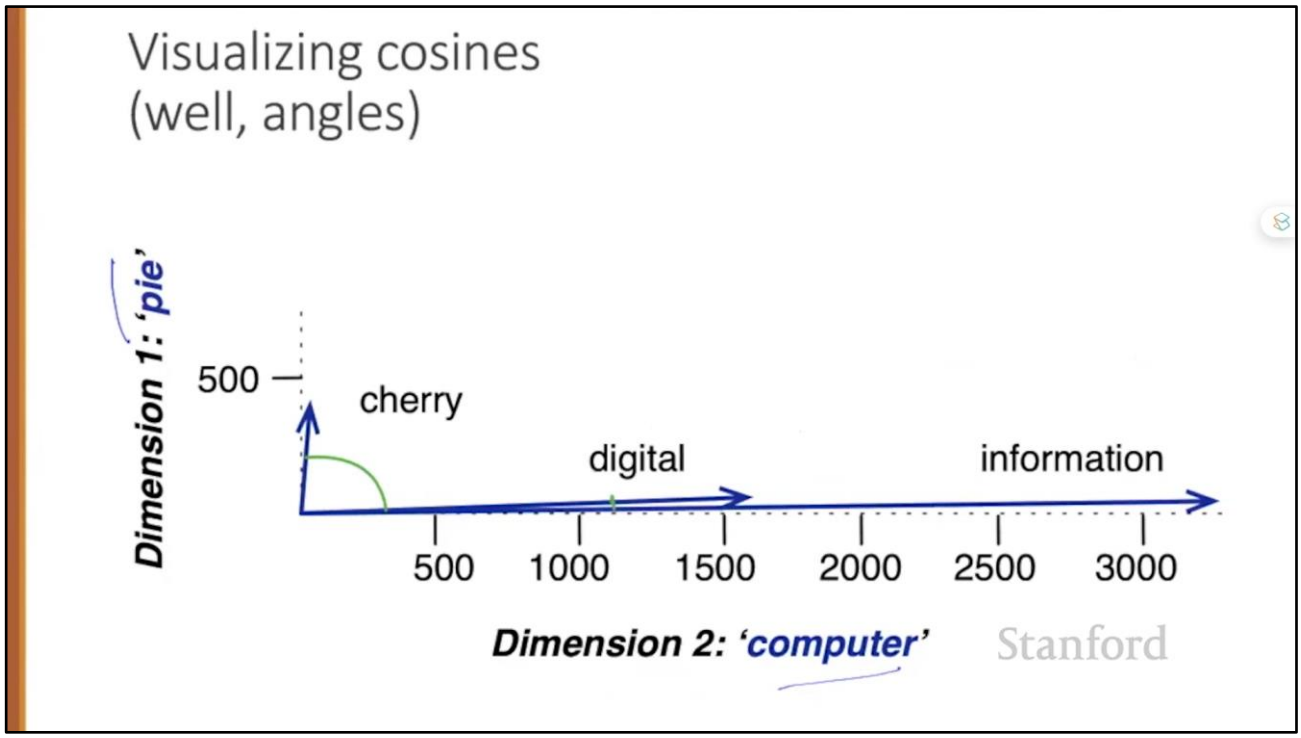
$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

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

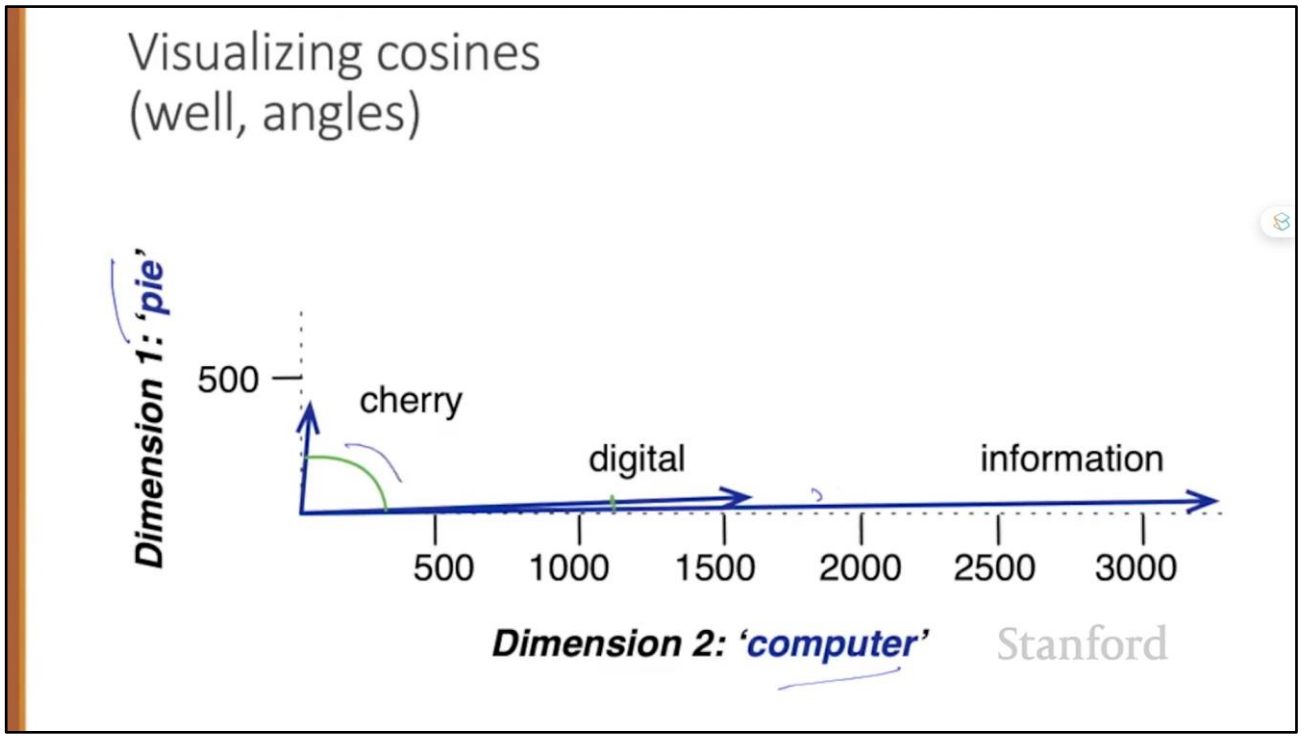
$$\cos(\text{cherry},\text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .017$$

$$\cos(\text{digital},\text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$

By contrast, the cosine between digital and information, we can compute the numbers the same way. But because both numbers are very high in the data dimension. And the computer dimension. We get a much higher cosine value. The model decides that information is way closer to digital than it is to cherry. A result that seems sensible. Here's a rough graphical demonstration of cosine similarity,



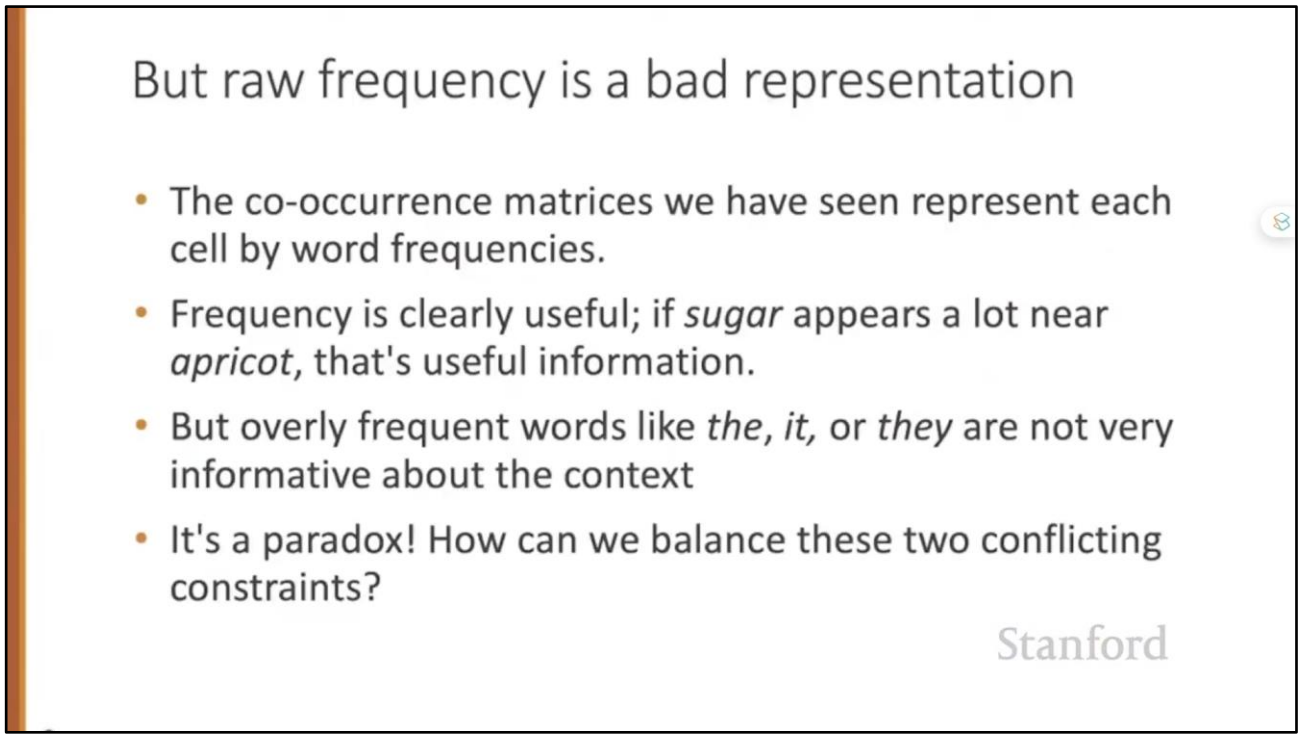
showing vectors for the words cherry digital and information in the tiny two dimensional space defined by counts of the words computer and pi nearby.



Note that the angle between digital and information is smaller than the angle between cherry and information.
When two vectors are more similar, the cosine is larger. But the angle is smaller. The cosine has its maximum of one when the angle between the two vectors is smallest and the coastline of all other angles is less than one. We've seen in detail the vector cosine, the most common similarity algorithm for two word vectors.

TF - IDF

Here we introduce tf-idf weighting, a common way to reweight counts in term document matrices.



The co-occurrence matrices we've been using to represent each cell by frequencies of words with other words or documents.
But it turns out that raw frequency is a very skewed metric and not very discriminative. So if sugar appears a lot near apricot, that's useful information.

But raw frequency is 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?

Stanford

But words like the or it which are very frequent and occur with all sorts of words. They're not very informative about any particular word. It's a bit of a paradox. How can we balance these two conflicting constraints?

Two common solutions for word weighting

tf-idf: tf-idf value for word t in document d:

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

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

PMI: (Pointwise mutual information)

- $\text{PMI}(w_1, w_2) = log \frac{p(w_1,w_2)}{p(w_1)p(w_2)}$

See if words like "good" appear more often with "great" than we would expect by chance

Stanford

There are two common solutions to this problem. The tf-idf algorithm usually used when the dimensions are documents and algorithms
based on point wise mutual information usually use when the dimensions are words. tf-idf makes use of a special weight called the inverse document frequency to down weight.

Two common solutions for word weighting

tf-idf: tf-idf value for word t in document d:

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

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

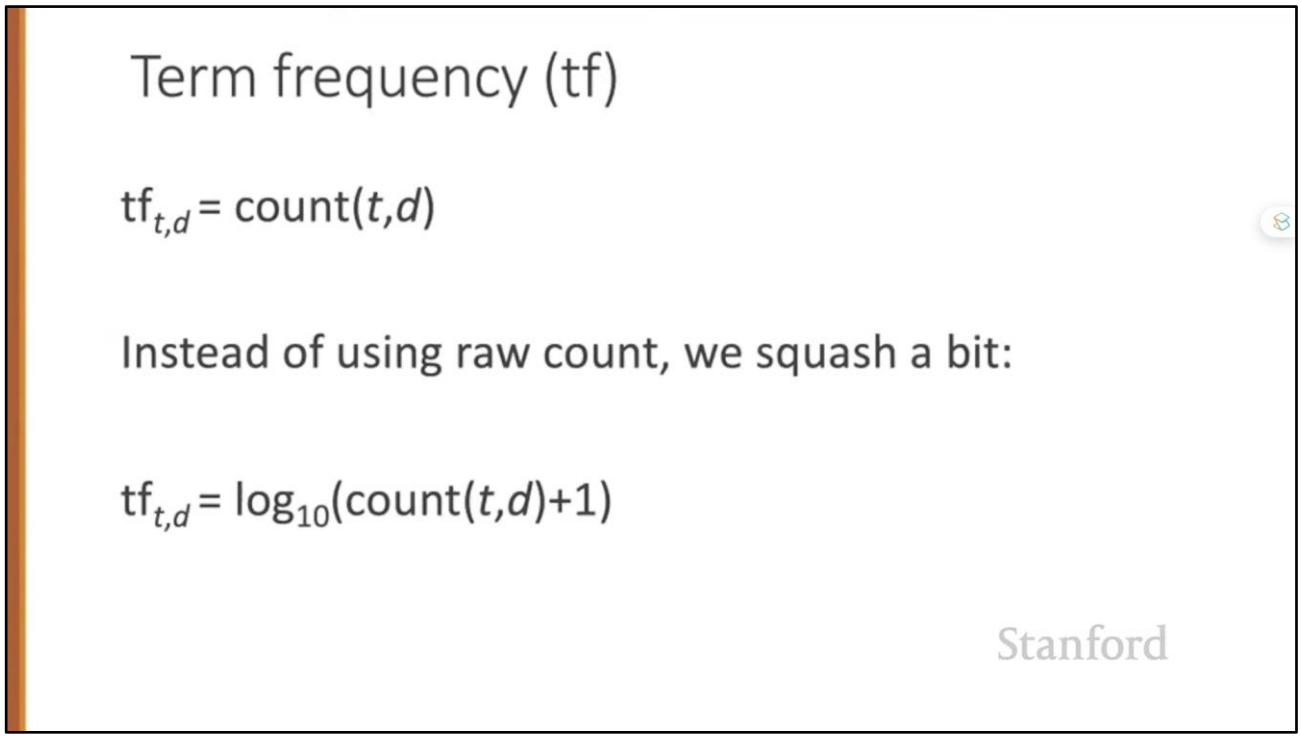
PMI: (Pointwise mutual information)

- $\text{PMI}(w_1, w_2) = log \frac{p(w_1,w_2)}{p(w_1)p(w_2)}$

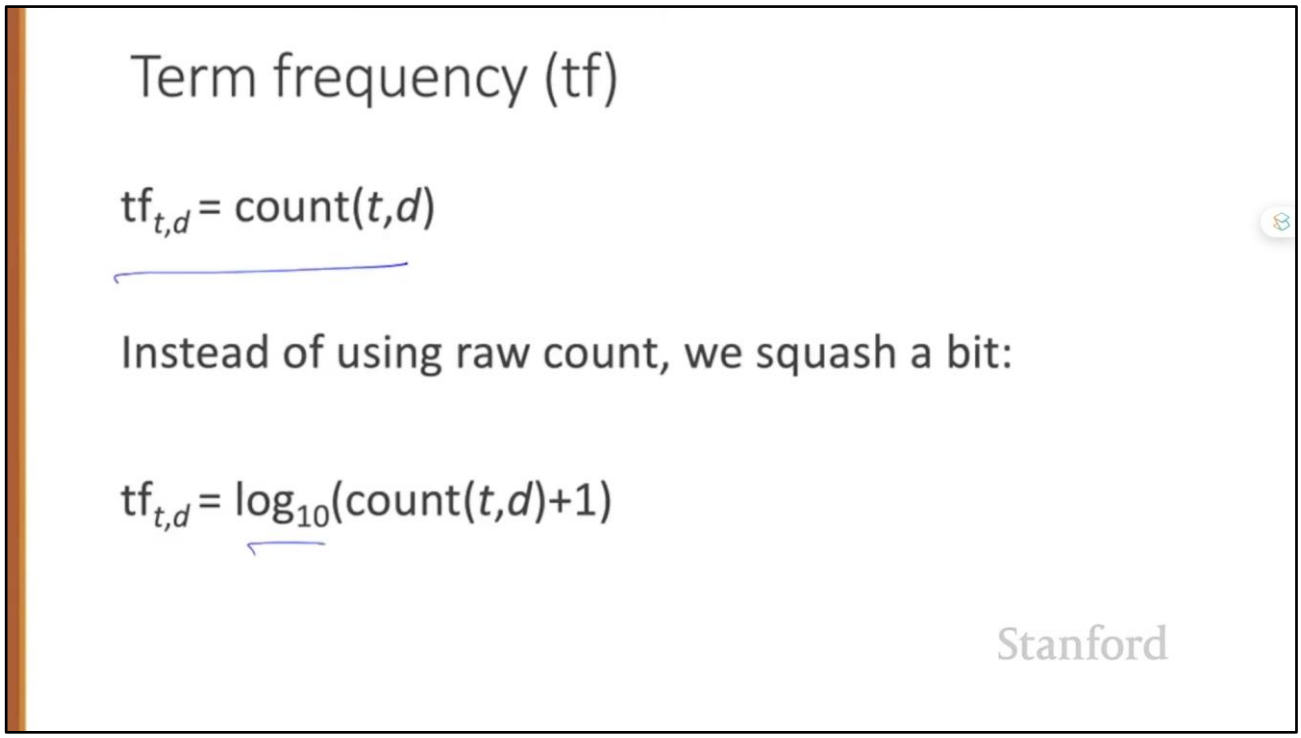
See if words like "good" appear more often with "great" than we would expect by chance

Stanford

Words like the or it. While PMI is a statistical measure that compares the probabilities we see with what we would have expected by chance.



The tf-idf algorithm is the product of two terms each term, capturing one of these two intuitions.
The first is the term frequency, just the frequency of the word T in the document D.
We could just use the raw count as the term frequency, but more commonly,



we squash the raw frequency a bit by using the log 10 of the frequency instead. The intuition is that a word appearing 100 times in a document doesn't make the word 100 times more likely to be relevant to the meaning of the document. And because we can't take the log of zero, we normally add one to the count.

Document frequency (df)

df_t *is* the number of documents *t* occurs in.

(note this is not collection frequency: total count across all documents)

"*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Stanford

The second factor in tf-idf is used to give a higher weight to words that occur only in a few documents, terms that are limited to a few documents are useful for discriminating.
Those documents from the rest of the collection terms that occur frequently across the entire collection aren't as helpful. And the document frequency of a term is the number of documents. It occurs in

Document frequency (df)

df_t *is* the number of documents *t* occurs in.

(note this is not collection frequency: total count across all documents)

"*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Stanford

It occurs in document frequency, by the way, is not the same as the collection frequency of a term, which is the total number of times the word appears in the whole collection in any document.
Considering the collection of Shakespeare 37 plays the two words Romeo and Action, the two words have identical collection frequency.
They both occur 113 times in all the plays, but very different document frequencies. Since Romeo only occurs in a single play.
So if our goal is to find documents about the romantic tribulations of Romeo, the word Romeo should be highly weighted, but not the word action.

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

We emphasize discriminative words like Romeo V, the inverse document frequency or IDF term weight.
The IDF is defined using the fraction "N over df" where N is the total number of documents in the collection and df_t is the number of documents in which term t occurs.

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

The IDF is defined using the fraction "N over df" where N is the total number of documents in the collection and df_t is the number of documents in which term t occurs.
The fewer documents in which the term occurs, the higher the weight, the lowest weight of one is assigned to terms that occur in all the documents.
Because of the large number of documents in many collections, this measure, too, is usually squashed with a log function.

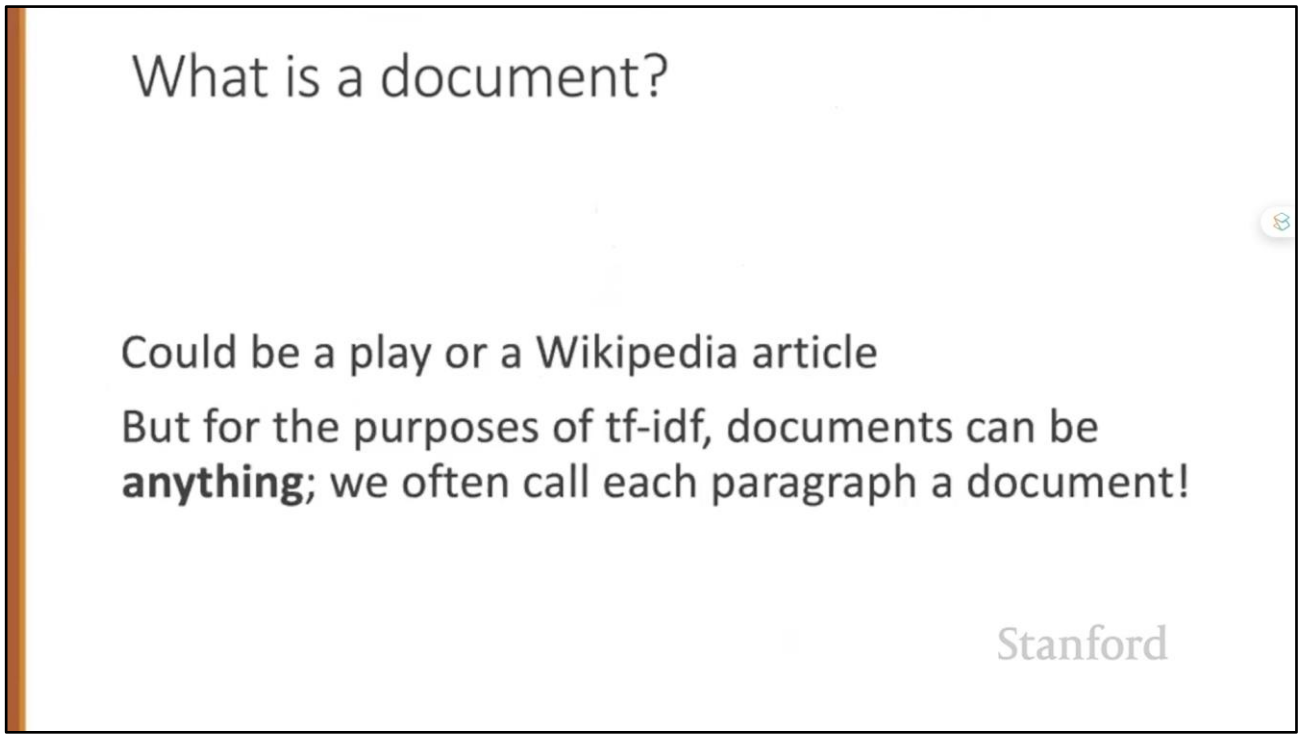
Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

The resulting definition for inverse document frequency, or IDF, is thus the log of N over df_t.
Here on the right are some idea of values for some words in the Shakespeare corpus, ranging from extremely informative words which occur only in one play like Romeo.
To those that occur in a few like salad or Falstaff, to those which are very common, like fool or so common as to be completely non discriminative since they occur in all 37 plays like good or sweet.



It's usually clear what counts as a document. In Shakespeare we'd use a play; when processing a collection of encyclopedia articles like Wikipedia, the document is a Wikipedia page.
In processing newspaper articles, the document is a single article. But very frequently you might need to break up the corpus and the documents yourself for the purposes of computing IDF because documents can be anything.
They don't have to be the original document. For example, we often treat each paragraph as a document which lets us compute tf-idf values even when we're dealing with a single document like a book.

Final tf-idf weighted value for a word

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

Stanford

So the tf-idf weighted value for a word T in Document D thus combines the term frequency of the term T in document T with the inverse document frequency of the term T.

Final tf-idf weighted value for a word

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

Raw counts:

	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

Stanford

Here's the raw counts in the Shakespeare term document matrix and now the tf-idf weighted version of the same matrix.

Final tf-idf weighted value for a word

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

Raw counts:

	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

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Note that the tf-idf values for the dimension corresponding to the word good have now all become zero.

Final tf-idf weighted value for a word

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

Raw counts:

	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

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Since this word appears in every document, the tf-idf algorithm leads it to be ignored.

Final tf-idf weighted value for a word

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

Raw counts:

	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

tf-idf:

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Similarly, the word fool, which appears in 36 out of the 37 plays, has a much lower weight than other words.
We've now seen how to use tf-idf weights to compute weighted vectors representing words.