



## Vector models of word meaning

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

*Firth (1957)*

# Why vector models of meaning?

## computing the similarity between words

“**fast**” is similar to “**rapid**”

“**tall**” is similar to “**height**”

Question answering:

*Q: “How **tall** is Mt. Everest?”*

*Candidate A: “The official **height** of Mount Everest is 29029 feet”*

# Word similarity for plagiarism detection

## MAINFRAMES

Mainframes **are primarily** referred to large computers with **rapid**, advanced processing capabilities that **can execute and** perform tasks **equivalent to many** Personal Computers (PCs) machines **networked together**. It is **characterized with high quantity** Random Access Memory (RAM), very large secondary storage devices, and **high-speed** processors to cater for the needs of the computers under its service.

**Consisting of** advanced components, mainframes have the capability of running multiple large applications required by **many and** most enterprises **and organizations**. **This is** one of its advantages. Mainframes are also suitable to cater for those applications **(programs)** or files that are of very **high**

## MAINFRAMES

Mainframes **usually are** referred those computers with **fast**, advanced processing capabilities that **could** perform **by itself** tasks **that may require a lot of** Personal Computers (PC) Machines. **Usually mainframes would have lots of** RAMs, very large secondary storage devices, and **very fast** processors to cater for the needs of those computers under its service.

**Due to the** advanced components mainframes have, **these computers** have the capability of running multiple large applications required by most enterprises, **which is** one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very **large** demand

# **Distributional models of meaning** **= vector-space models of meaning** **= vector semantics**

**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.”

Firth (1957):

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

# Intuition of distributional word similarity

- Nida example:

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

- From context words humans can guess **tesgüino** means
  - an alcoholic beverage like **beer**
- Intuition for algorithm:
  - Two words are similar if they have similar word contexts.

# Three kinds of vector models

## Sparse vector representations

1. Mutual-information weighted word co-occurrence matrices

## Dense vector representations:

2. Singular value decomposition (and Latent Semantic Analysis)
3. Neural-network-inspired models (skip-grams, CBOW)

# Shared intuition

- Model the meaning of a word by “embedding” in a vector space.
- The meaning of a word is a vector of numbers
  - Vector models are also called “**embeddings**”.
- Contrast: word meaning is represented in many computational linguistic applications by a vocabulary index (“word number 545”)

# Term-document matrix

- Each cell: count of term  $t$  in a document  $d$ :  $tf_{t,d}$ 
  - Each document is a **count vector** in  $\mathbb{N}^v$ : a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0



# Term-document matrix

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

# The words in a term-document matrix

- Each word is a count vector in  $\mathbb{N}^D$ : a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

# The words in a term-document matrix

- Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

# Term-context matrix for word similarity

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

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

# The word-word or word-context matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of  $\pm 4$  words
- A word is now defined by a vector over counts of context words
- Instead of each vector being of length  $D$
- Each vector is now of length  $|V|$
- The word-word matrix is  $|V| \times |V|$

# Word-Word matrix

## Sample contexts $\pm 7$ words

sugar, a sliced lemon, a tablespoonful of  
their enjoyment. Cautiously she sampled her first  
well suited to programming on the digital  
for the purpose of gathering data and

**apricot**  
**pineapple**  
**computer.**  
**information**

preserve or jam, a pinch each of,  
and another fruit whose taste she likened  
In finding the optimal R-stage policy from  
necessary for the study authorized in the

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	
...	...						

# Word-word matrix

- We showed only 4x6, but the real matrix is 50,000 x 50,000
  - So it's very **sparse**
    - Most values are 0.
  - That's OK, since there are lots of efficient algorithms for sparse matrices.
- The size of windows depends on your goals
  - The shorter the windows , the more **syntactic** the representation  
 $\pm 1-3$
  - The longer the windows, the more **semantic** the representation  
 $\pm 4-10$

# 2 kinds of co-occurrence between 2 words

(Schütze and Pedersen, 1993)

- First-order co-occurrence (**syntagmatic association**):
  - They are typically nearby each other.
  - *wrote* is a first-order associate of *book* or *poem*.
- Second-order co-occurrence (**paradigmatic association**):
  - They have similar neighbors.
  - *wrote* is a second- order associate of words like *said* or *remarked*.



# Vector Semantics

Positive Pointwise Mutual  
Information (PPMI)

# Problem with raw counts

- Raw word frequency is not a great measure of association between words
  - It's very skewed
    - “the” and “of” are very frequent, but maybe not the most discriminative
- We'd rather have a measure that asks whether a context word is **particularly informative** about the target word.
  - Positive Pointwise Mutual Information (PPMI)

# Pointwise Mutual Information

## Pointwise mutual information:

Do events  $x$  and  $y$  co-occur more than if they were independent?

$$\text{PMI}(X, Y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

## PMI between two words: (Church & Hanks 1989)

Do words  $x$  and  $y$  co-occur more than if they were independent?

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}$$

# Positive Pointwise Mutual Information

- PMI ranges from  $-\infty$  to  $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine  $w_1$  and  $w_2$  whose probability is each  $10^{-6}$
    - Hard to be sure  $p(w_1, w_2)$  is significantly different than  $10^{-12}$
  - Plus it's not clear people are good at “unrelatedness”
- So we just replace negative PMI values by 0
- Positive PMI (PPMI) between word1 and word2:

$$\text{PPMI}(\text{word}_1, \text{word}_2) = \max\left(\log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}, 0\right)$$

# Computing PPMI on a term-context matrix

- Matrix  $F$  with  $W$  rows (words) and  $C$  columns (contexts)
- $f_{ij}$  is # of times  $w_i$  occurs in context  $c_j$

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{i*} = \frac{\sum_{j=1}^C f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}} \quad p_{*j} = \frac{\sum_{i=1}^W f_{ij}}{\sum_{i=1}^W \sum_{j=1}^C f_{ij}}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*} p_{*j}} \quad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

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

apricot  
pineapple  
digital  
information

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

$$p(w=\text{information}, c=\text{data}) = 6/19 = .32$$

$$p(w=\text{information}) = 11/19 = .58$$

$$p(c=\text{data}) = 7/19 = .37$$

$$p(w_i) = \frac{\sum_{j=1}^C f_{ij}}{N}$$

$$p(c_j) = \frac{\sum_{i=1}^W f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
<b>p(context)</b>	0.16	0.37	0.11	0.26	0.11	

		p(w,context)					p(w)
		computer	data	pinch	result	sugar	
$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$	apricot	0.00	0.00	0.05	0.00	0.05	0.11
	pineapple	0.00	0.00	0.05	0.00	0.05	0.11
	digital	0.11	0.05	0.00	0.05	0.00	0.21
	information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)		0.16	0.37	0.11	0.26	0.11	

- $pmi(\text{information}, \text{data}) = \log_2 (.32 / (.37 * .58)) = .58$

*(.57 using full precision)*

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

# Weighting PMI

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values
- Two solutions:
  - Give rare words slightly higher probabilities
  - Use add-one smoothing (which has a similar effect)



# Weighting PMI: Giving rare context words slightly higher probability (Levy et al., 2015)

- Raise the context probabilities to  $\alpha = 0.75$ :

$$\text{PPMI}_{\alpha}(w, c) = \max(\log_2 \frac{P(w, c)}{P(w)P_{\alpha}(c)}, 0)$$

$$P_{\alpha}(c) = \frac{\text{count}(c)^{\alpha}}{\sum_c \text{count}(c)^{\alpha}}$$

- This helps because  $P_{\alpha}(c) > P(c)$  for rare  $c$
- Consider two events,  $P(a) = .99$  and  $P(b) = .01$

$$P_{\alpha}(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97 \quad P_{\alpha}(b) = \frac{.01^{.75}}{.01^{.75} + .99^{.75}} = .03$$

# Laplace smoothing

	Add-2					Smoothed Count(w,context)
	computer	data	pinch	result	sugar	
apricot	2	2	3	2	3	
pineapple	2	2	3	2	3	
digital	4	3	2	3	2	
information	3	8	2	6	2	

	$p(w,context) \frac{1}{\text{add-2}}$					$p(w)$
	computer	data	pinch	result	sugar	
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
$p(context)$	0.19	0.25	0.17	0.22	0.17	

# PPMI versus add-2 smoothed PPMI

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

	PPMI(w,context)[add-2]				
	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

# Vector Semantics

Measuring similarity: the  
cosine

# Measuring similarity

- Given 2 target words  $v$  and  $w$
- We'll need a way to measure their similarity.
- Most measures of vector similarity are based on the:
- **Dot product** or **inner product** from linear algebra

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

- High when two vectors have large values in same dimensions.
- Low (in fact 0) for **orthogonal vectors** with zeros in complementary distribution

# Problem with dot product

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

- Dot product is higher if the vector is longer. Vector length:

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

- Vectors are longer if they have higher values in each dimension
- That means more frequent words will have higher dot products
- That's bad: we don't want a similarity metric to be sensitive to word frequency

## Solution: cosine

- Just divide the dot product by the length of the two vectors!

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|}$$

- This turns out to be the cosine of the angle between them!

$$\vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos \theta$$

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

# Cosine for computing similarity

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

$v_i$  is the PPMI value for word  $v$  in context  $i$

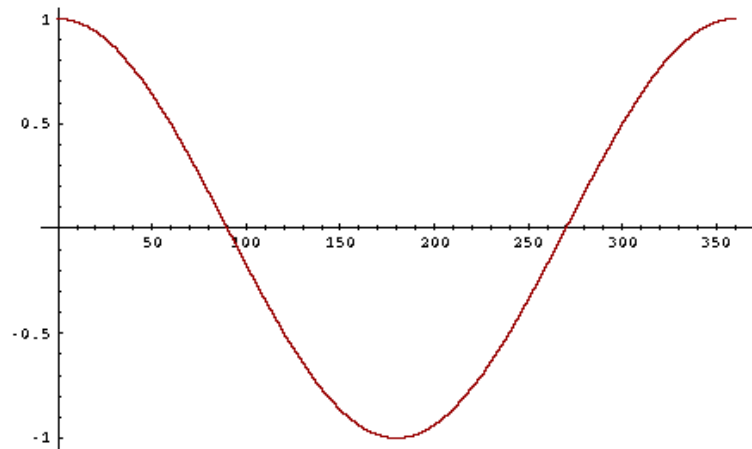
$w_i$  is the PPMI value for word  $w$  in context  $i$ .

$\text{Cos}(\vec{v}, \vec{w})$  is the cosine similarity of  $\vec{v}$  and  $\vec{w}$



# Cosine as a similarity metric

- -1: vectors point in opposite directions
  - +1: vectors point in same directions
  - 0: vectors are orthogonal
- 
- Raw frequency or PPMI are non-negative, so cosine range 0-1



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

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

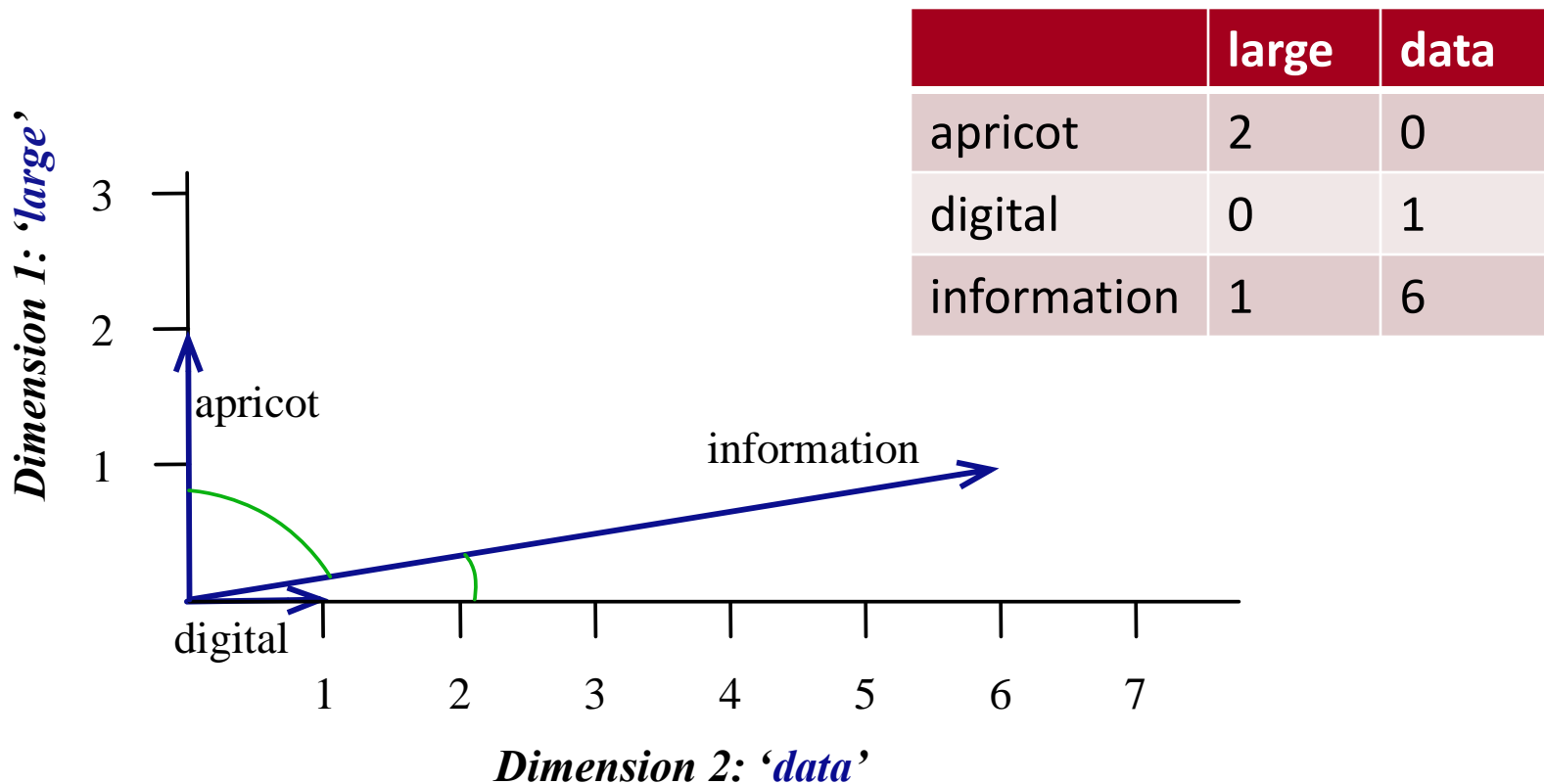
Which pair of words is more similar?

$$\text{cosine}(\text{apricot}, \text{information}) = \frac{2 + 0 + 0}{\sqrt{2 + 0 + 0} \sqrt{1 + 36 + 1}} = \frac{2}{\sqrt{2} \sqrt{38}} = .23$$

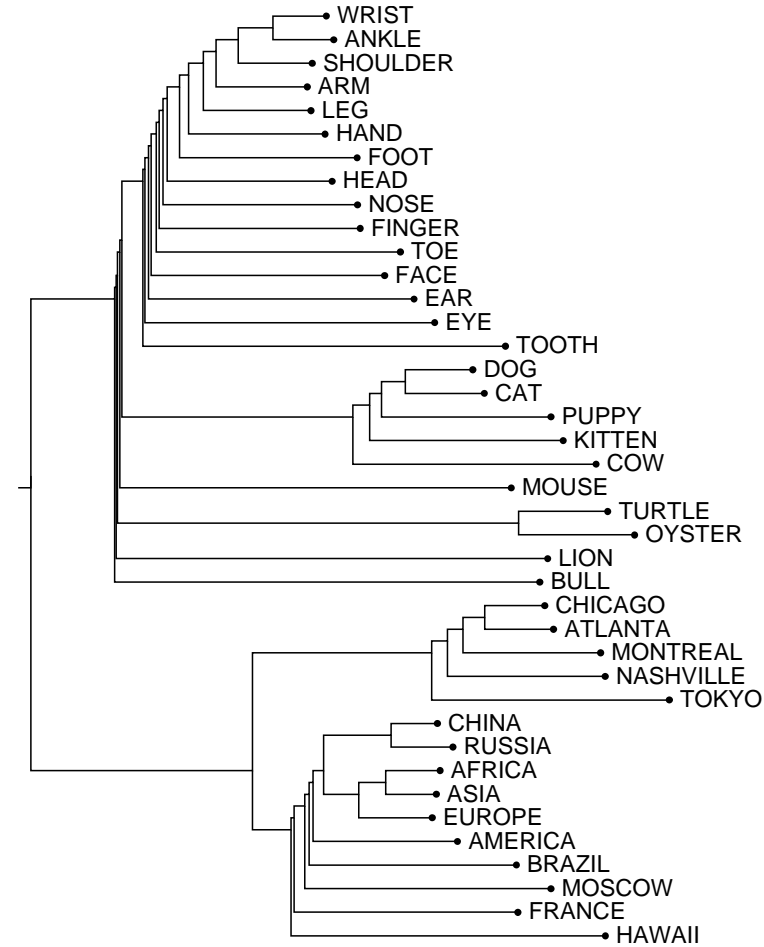
$$\text{cosine}(\text{digital}, \text{information}) = \frac{0 + 6 + 2}{\sqrt{0 + 1 + 4} \sqrt{1 + 36 + 1}} = \frac{8}{\sqrt{5} \sqrt{38}} = .58$$

$$\text{cosine}(\text{apricot}, \text{digital}) = \frac{0 + 0 + 0}{\sqrt{1 + 0 + 0} \sqrt{0 + 1 + 4}} = 0$$

# Visualizing vectors and angles



# Clustering vectors to visualize similarity in co-occurrence matrices



Rohde et al. (2006)

## Other possible similarity measures

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

$$\text{sim}_{\text{Jaccard}}(\vec{v}, \vec{w}) = \frac{\sum_{i=1}^N \min(v_i, w_i)}{\sum_{i=1}^N \max(v_i, w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v}, \vec{w}) = \frac{2 \times \sum_{i=1}^N \min(v_i, w_i)}{\sum_{i=1}^N (v_i + w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v} || \vec{w}) = D(\vec{v} | \frac{\vec{v} + \vec{w}}{2}) + D(\vec{w} | \frac{\vec{v} + \vec{w}}{2})$$

# Using syntax to define a word's context

- Zellig Harris (1968)

“The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities”
- **Two words are similar if they have similar syntactic contexts**

**Duty** and **responsibility** have similar syntactic distribution:

**Modified by  
adjectives**

additional, administrative, assumed, collective,  
congressional, constitutional ...

**Objects of verbs**

assert, assign, assume, attend to, avoid, become, breach..

# Co-occurrence vectors based on syntactic dependencies

Dekang Lin, 1998 “Automatic Retrieval and Clustering of Similar Words”

- Each dimension: a context word in one of R grammatical relations
  - Subject-of- “absorb”
- Instead of a vector of  $|V|$  features, a vector of  $R/V$
- Example: counts for the word *cell* :

	subj-of, absorb	subj-of, adapt	subj-of, behave	...	pobj-of, inside	pobj-of, into	...	nmod-of, abnormality	nmod-of, anemia	nmod-of, architecture	...	obj-of, attack	obj-of, call	obj-of, come from	obj-of, decorate	...	nmod, bacteria	nmod, body	nmod, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

# Syntactic dependencies for dimensions

- Alternative (Padó and Lapata 2007):
  - Instead of having a  $|V| \times R|V|$  matrix
  - Have a  $|V| \times |V|$  matrix
  - But the co-occurrence counts aren't just counts of words in a window
  - But counts of words that occur in one of  $R$  dependencies (subject, object, etc).
  - So  $M(\text{"cell"}, \text{"absorb"}) = \text{count}(\text{subj}(\text{cell}, \text{absorb})) + \text{count}(\text{obj}(\text{cell}, \text{absorb})) + \text{count}(\text{pobj}(\text{cell}, \text{absorb}))$ , etc.



# PMI applied to dependency relations

Hindle, Don. 1990. Noun Classification from Predicate-Argument Structure. ACL

Object of “drink”	Count	PMI
tea	2	11.8
liquid	2	10.5
wine	2	9.3
anything	3	5.2
it	3	1.3

- “Drink it” more common than “drink wine”
- But “wine” is a better “drinkable” thing than “it”

# Alternative to PPMI for measuring association

- **tf-idf** (that's a hyphen not a minus sign)
- The combination of two factors
  - **Term frequency** (Luhn 1957): frequency of the word (can be logged)
  - **Inverse document frequency** (IDF) (Sparck Jones 1972)
    - $N$  is the total number of documents
    - $df_i$  = "document frequency of word  $i$ "
    - $=$  # of documents with word  $i$
- $w_{ij}$  = word  $i$  in document  $j$

$$idf_i = \log \frac{N}{df_i}$$

$$w_{ij} = tf_{ij} idf_i$$

# tf-idf not generally used for word-word similarity

- But is by far the most common weighting when we are considering the relationship of words to documents

# Vector Semantics

Evaluating similarity

# Evaluating similarity

- Extrinsic (task-based, end-to-end) Evaluation:
  - Question Answering
  - Spell Checking
  - Essay grading
- Intrinsic Evaluation:
  - Correlation between algorithm and human word similarity ratings
    - Wordsim353: 353 noun pairs rated 0-10.  $sim(plane, car)=5.77$
  - Taking TOEFL multiple-choice vocabulary tests
    - Levied is closest in meaning to:  
imposed, believed, requested, correlated

# Summary

- Distributional (vector) models of meaning
  - **Sparse** (PPMI-weighted word-word co-occurrence matrices)
  - **Dense:**
    - Word-word SVD 50-2000 dimensions
    - Skip-grams and CBOW