ARTIFICIAL INTELLIGENCE: METHODS & APPLICATIONS
INTRODUCTION TO VECTOR SEMANTICS / WORD EMBEDDING

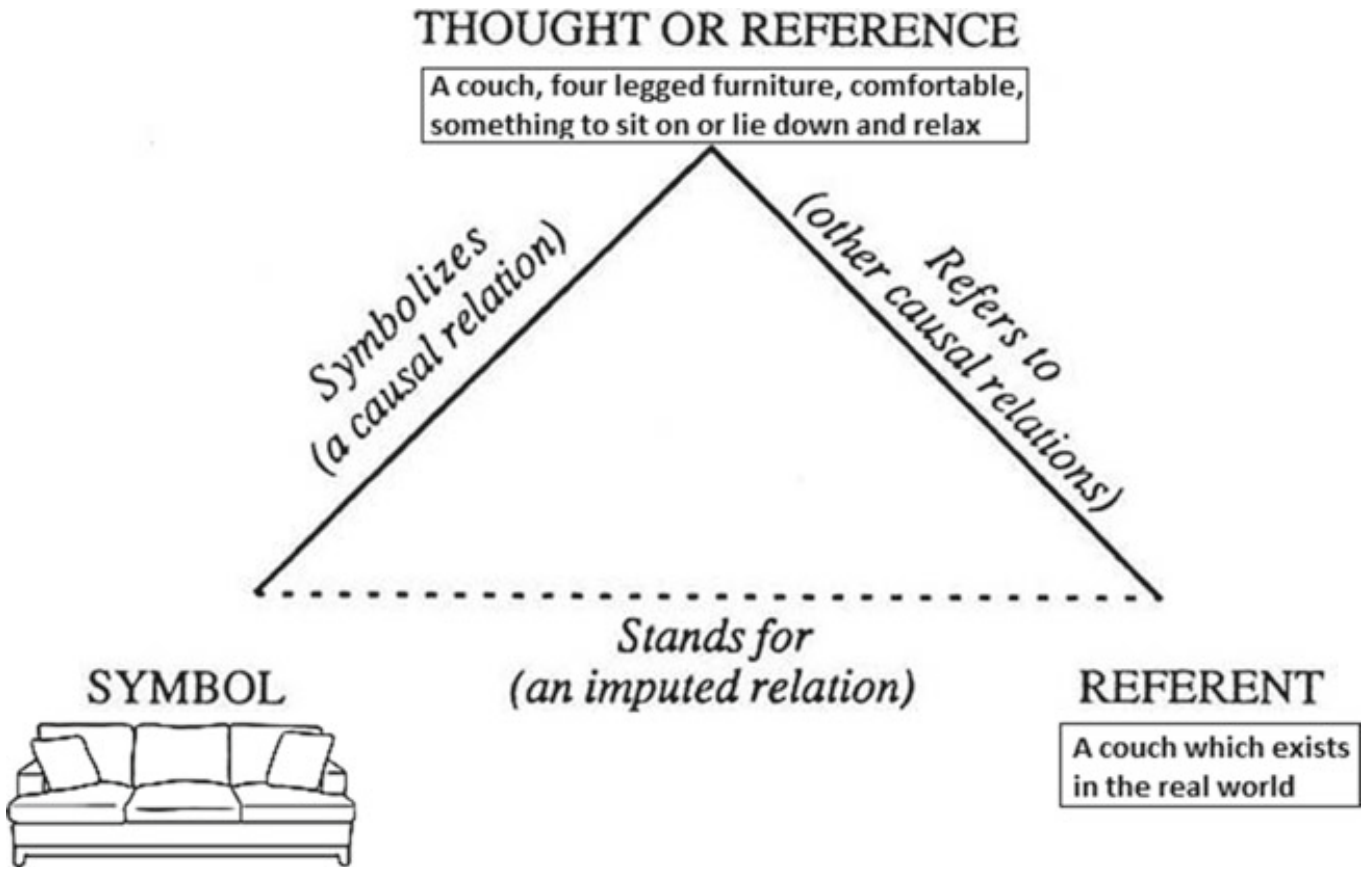
Plamen Petrov

ppetro2@uic.edu
January 25, 2018

The relationship between language and reality

The “Triangle of Reference” model a.k.a. “meaning of meaning model”

Proposed by Charles Ogden and Ivor Richards in 1923



Adapted from “Text Analytics”, Dipanjan Sarkar

Vector Semantics

The Vector Space Model

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"*

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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

Words that occur in similar contexts tend to have similar meanings.

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!"

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Distributional Methods and Vector Semantics

- ***Distributional methods*** - the meaning of a word is computed from the distribution of words around it.
- These words are generally represented as a vector or array of numbers related in some way to counts, and so these methods are often called ***vector semantics***.
- A simple method in which the meaning of a word is simply defined by how often it occurs near other words.
- This method results in very long ('high dimensional') vectors that are sparse, i.e. contain mostly zeros (since most words never occur in the context of others).
- Then expand on this simple idea by introducing three ways of constructing short, dense vectors that have useful semantic properties.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Word Embedding

- The shared intuition of vector space models of semantics is to model a word by embedding it into a vector space.
- For this reason the representation of a word as a vector is often called an ***embedding***.
- By contrast, in many traditional NLP applications, a word is represented as an index in a vocabulary list, or as a string of letters.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Vector Models and Natural Language Semantics

- Vector models of meaning have been used in NLP for over 50 years.
- They are commonly used as features to represent words in applications from named entity extraction to parsing to semantic role labeling to relation extraction.
- Vector models are also the most common way to compute semantic similarity, the similarity between two words, two sentences, or two documents, an important tool in practical applications like:
  - Question Answering
  - Summarization
  - Automatic essay grading.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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 demand by its users (clients). Examples of such organizations and enterprises using mainframes are online shopping websites such as Ebay, Amazon, and computing-giant

MAINFRAMES

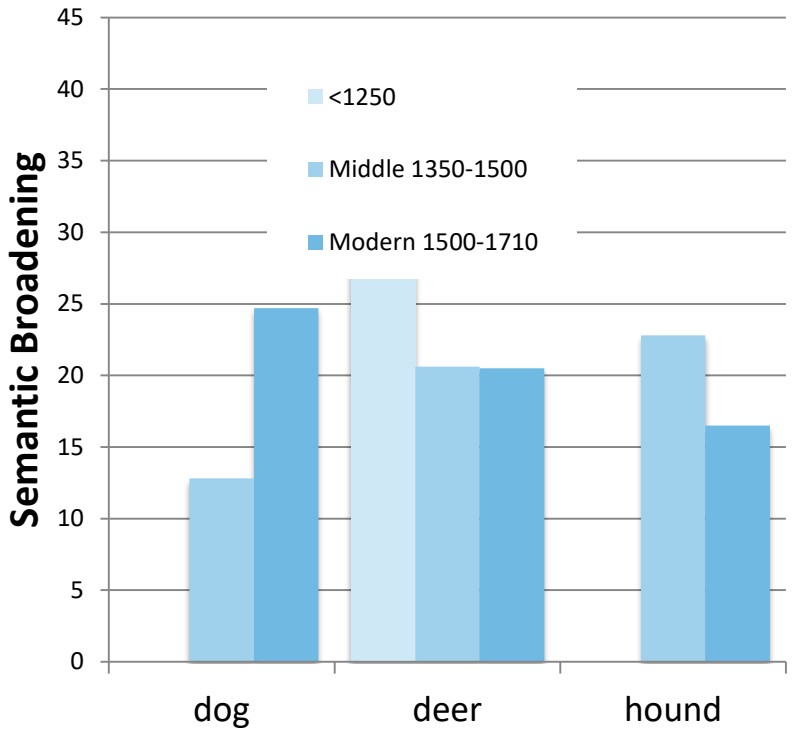
Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater 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 by its users (clients). Examples of these include the large online shopping websites -i.e. : Ebay, Amazon, Microsoft, etc.

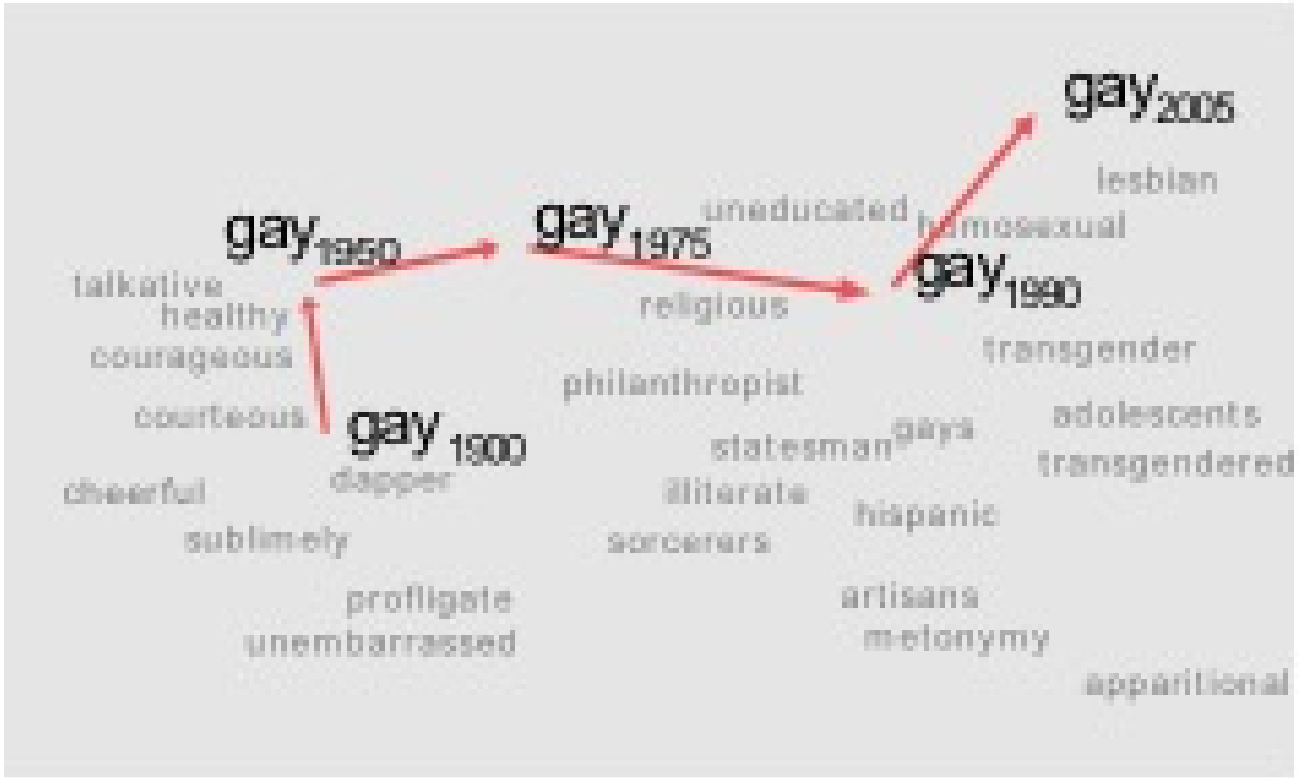
Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Word similarity for historical linguistics: semantic change over time

Sagi, Kaufmann Clark 2013



Kulkarni, Al-Rfou, Perozzi, Skiena 2015



Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Four 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)
4. Brown clusters

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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”)
- Old philosophy joke:

  Q: What’s the meaning of life?

  A: LIFE’

Adapted from “Speech and Language Processing”, Daniel Jurafsky, James H. Martin

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

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Term-Document Matrix

	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	5	117	0	0

Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

In a ***term-document matrix***, each row represents a word in the vocabulary and each column represents a document from some collection.
Example: a small selection from a term-document matrix showing the occurrence of four words in 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 clown appeared 117 times in Twelfth Night.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

Term-Document Matrix

	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	5	117	0	0

Figure 15.2 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each document is represented as a column vector of length four.

The term-document matrix was first defined as part of the ***vector space model*** of information retrieval.
In the ***vector space model***, a document is represented as a count vector e.g. a column.
So ***As You Like It*** is represented as the list **[1,2,37,5]** and ***Julius Caesar*** is represented as the list **[8,12,1,0]**.
A vector space is a collection of vectors, characterized by their dimension.
The ordering of the numbers in a vector space is not arbitrary; each position indicates a meaningful dimension on which the documents can vary.
Thus the ***first dimension*** for both these vectors corresponds to the number of times the word **battle** occurs, and we can compare each dimension, noting for example that the vectors for ***As You Like It*** and ***Twelfth Night*** have the same value 1 for the first dimension.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

The Vector Space Model

- We can think of the vector for a document as identifying a point in |V|-dimensional space.
- Thus the Shakespeare documents are points in 4-dimensional space.
- Since 4- dimensional spaces are hard to draw in textbooks, the following figure shows a visualization in two dimensions, where we have arbitrarily chosen the dimensions corresponding to the words battle and fool.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

The Vector Space Model

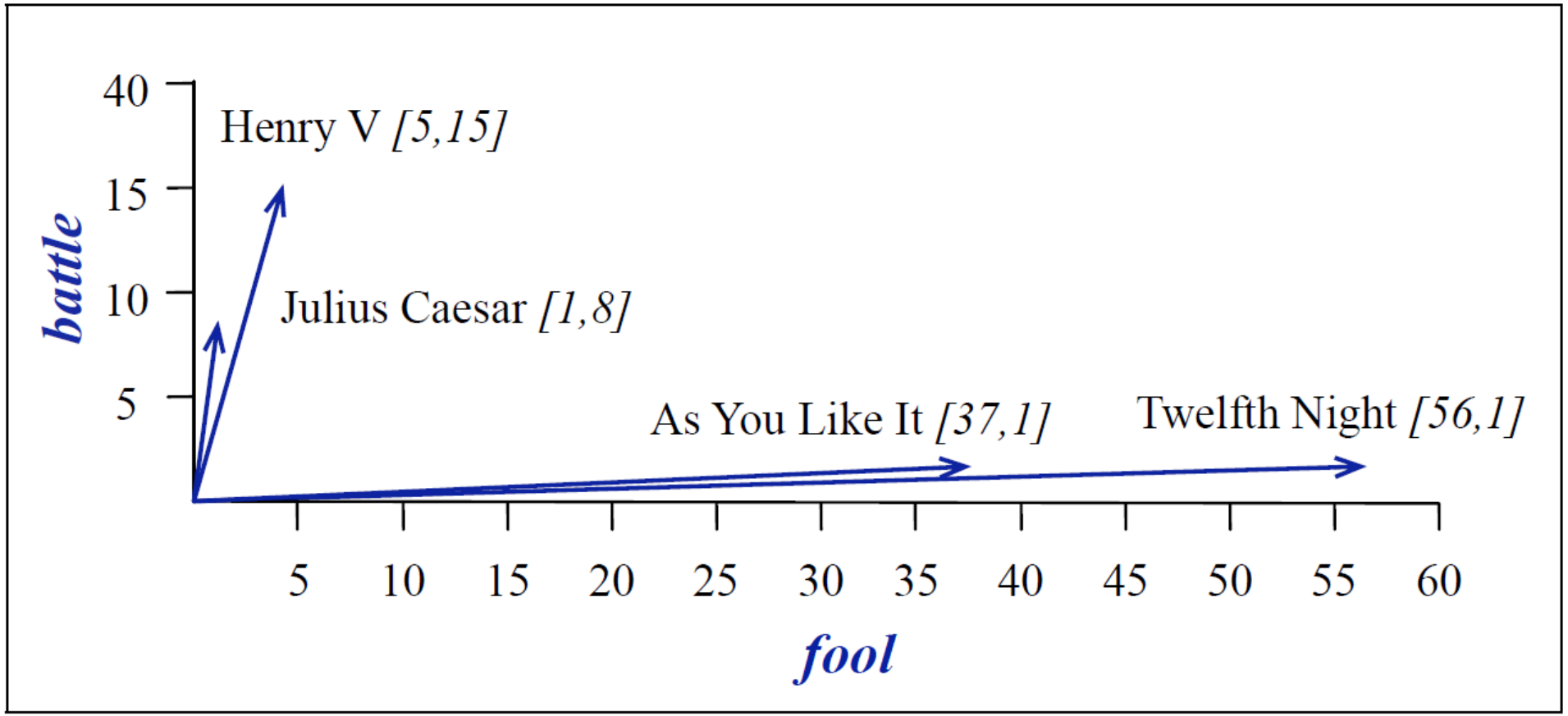


Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

The Vector Space Model

Term-document matrices were originally defined as a means of finding similar documents for the task of document **information retrieval**. Two documents that are similar will tend to have similar words, and if two documents have similar words their column vectors will tend to be similar. The vectors for the comedies *As You like It* [1,2,37,5] and *Twelfth Night* [1,2,58,117] look a lot more like each other (more fools and clowns than soldiers and battles) than they do like *Julius Caesar* [8,12,1,0] or *Henry V* [15,36,5,0].

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 Vector Space Model

We can see the intuition with the raw numbers; in the first dimension (battle) the comedies have low numbers and the others have high numbers (we see it visually); we'll also need to quantify this intuition more formally.

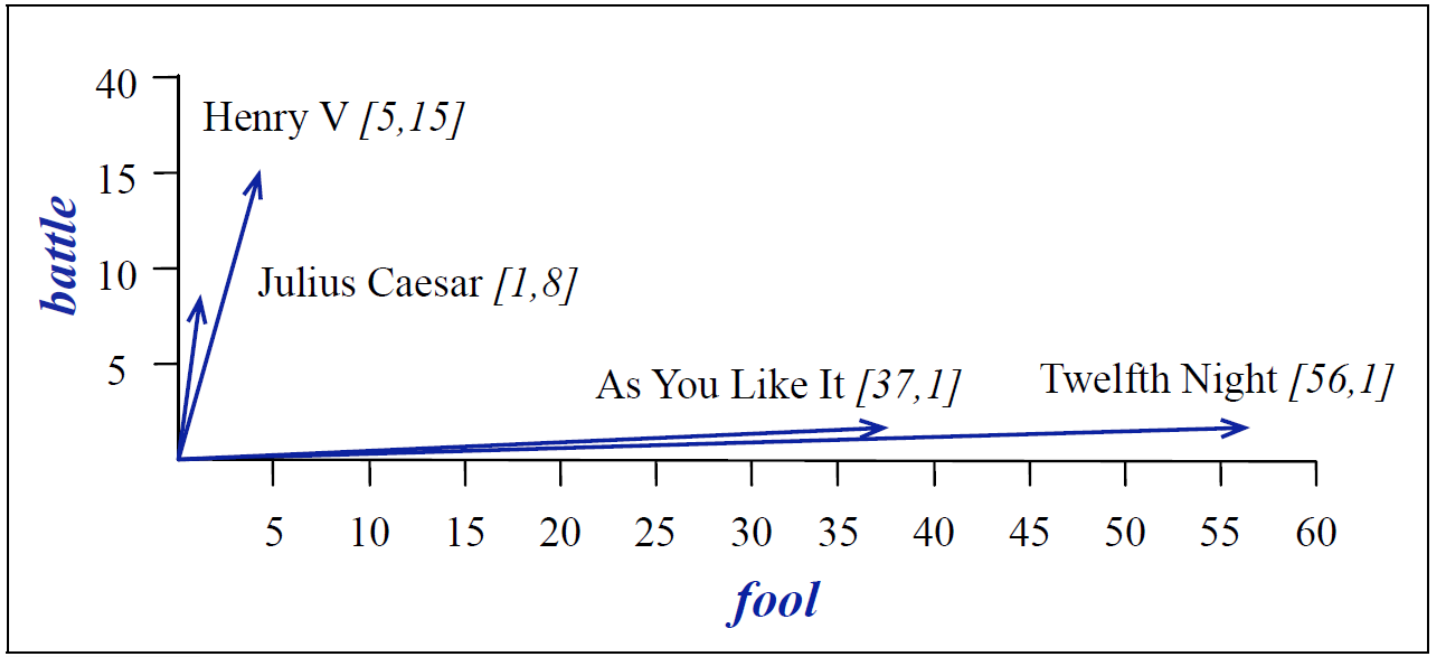


Figure 15.3 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

The Vector Space Model

A real term-document matrix wouldn't just have 4 rows and columns.

More generally, the term-document matrix X has:
- |V| rows (one for each word type in the vocabulary) and
- D columns (one for each document in the collection);

Vocabulary sizes are generally at least in the tens of thousands, and the number of documents can be enormous (think about all the pages on the web).

The Vector Space Model and Information Retrieval

A **Information retrieval (IR)** is the task of finding the document **d** from the **D** documents in some collection that best matches a query **q**.

For IR we'll therefore also represent a query by a vector, also of length |V|, and we'll need a way to compare two vectors to find how similar they are.

Doing IR will also require efficient ways to store and manipulate these vectors, which is accomplished by making use of the convenient fact that these vectors are sparse, i.e., mostly zeros.

Some components of the vector comparison process: the tf-idf term weighting, and the cosine similarity metric.

The Vector Space Model – Words as Vectors

- Documents can be represented as vectors in a vector space.
- Vector semantics can also be used to represent the meaning of words
  - ✓ Associating each word with a vector.
- The word vector is now a row vector rather than a column vector, and hence
- the dimensions of the vector are different.
- The four dimensions of the vector for ***fool***, [37,58,1,5], correspond to the four Shakespeare plays.
- The same four dimensions are used to form the vectors for the other 3 words:
  - ✓ clown, [5, 117, 0, 0];
  - ✓ battle, [1,1,8,15]; and
  - ✓ soldier [2,2,12,36].
- Each entry in the vector thus represents the counts of the word's occurrence in the document corresponding to that dimension.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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 Vector Space Model – Words as Vectors

- For documents:
  - ✓ similar documents had similar vectors, because similar documents tend to have similar words.
- This same principle applies to words:
  - ✓ similar words have similar vectors because they tend to occur in similar documents.
- The term-document matrix lets us represent the meaning of a word by the documents it tends to occur in.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

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

The Vector Space Model – Words as Vectors

- It is most common to use a different kind of context for the dimensions of a word's vector representation.
- Rather than the **term-document matrix** we use the **term-term matrix**, more commonly called the **word-word matrix** or the **term context matrix**, in which the columns are labeled by words rather than documents.
- This matrix is thus of dimensionality |V|x|V| and each ***cell*** records the **number of times** the ***row (target) word*** and the ***column (context) word*** co-occur in some context in some training corpus.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

The Vector Space Model – Words as Vectors

- The context could be the document, in which case the cell represents the number of times the two words appear in the same document.
- It is most common, however, to use smaller contexts, generally a window around the word, for example of 4 words to the left and 4 words to the right.
- In this case the cell represents the number of times (in some training corpus) the column word occurs in such a (+/-)-4 word window around the row word.

Adapted from "Speech and Language Processing", Daniel Jurafsky, James H. Martin

The Vector Space Model

	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	

Figure 15.4 Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions (hand-picked for pedagogical purposes). The vector for the word *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

The Vector Space Model

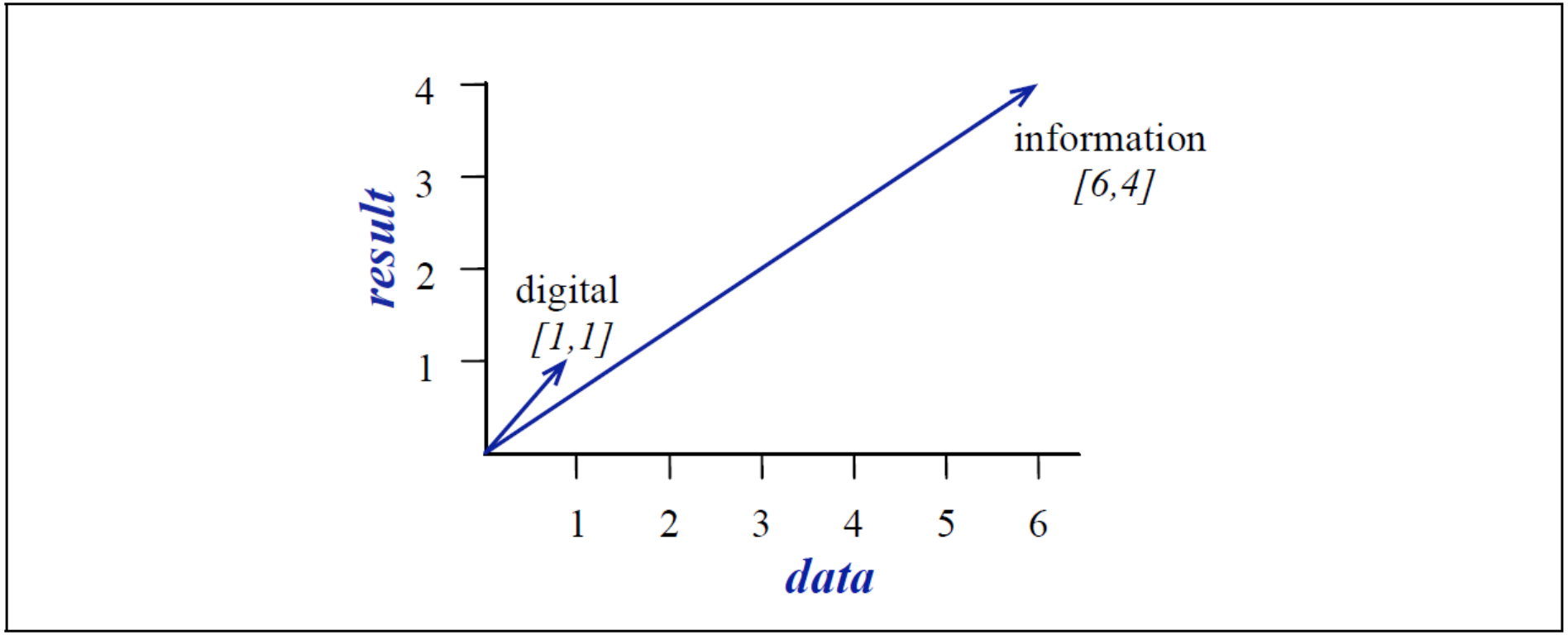


Figure 15.5 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *result*.

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|x|V|

Word-Word matrix
Sample contexts ± 7 words (from Brown corpus)

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** 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	
…	…						

- The two words apricot and pineapple are more similar to each other (both pinch and sugar tend to occur in their window) than they are to other words like digital;
- Conversely, digital and information are more similar to each other than, say, to apricot.

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

    ± 1-3 very syntactic
  - The longer the windows, the more **semantic** the representation

    ± 4-10 more semantic

The Vector Space Model

Note that $|V|$, the length of the vector, is generally the size of the vocabulary, usually between 10,000 and 50,000 words (using the most frequent words in the training corpus; keeping words after about the most frequent 50,000 or so is generally not helpful). But of course since most of these numbers are zero these are **sparse** vector representations, and there are efficient algorithms for storing and computing with sparse matrices.

The size of the window used to collect counts can vary based on the goals of the representation, but is generally between 1 and 8 words on each side of the target word (for a total context of 3-17 words). In general, the shorter the window, the more syntactic the representations, since the information is coming from immediately nearby words; the longer the window, the more semantic the relations.

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

The Vector Space Model

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)

- Based on the mutual information between two random variables X and Y:

$$I(X,Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

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?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Pointwise Mutual Information

The **pointwise mutual information** (Fano, 1961)[1] is a measure of how often two events x and y occur, compared with what we would expect if they were independent:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{15.3}$$

We can apply this intuition to co-occurrence vectors by defining the pointwise mutual information association between a target word w and a context word c as

$$\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)} \tag{15.4}$$

The numerator tells us how often we observed the two words together (assuming we compute probability by using the MLE). The denominator tells us how often we would **expect** the two words to co-occur assuming they each occurred independently, so their probabilities could just be multiplied. Thus, the ratio gives us an estimate of how much more the target and feature co-occur than we expect by chance.

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 w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) 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:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

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

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

$$\text{PPMI}_{ij} = \max(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0)$$

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

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

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=information,c=data) = 6/19 = .32

p(w=information) = 11/19 = .58

p(c=data) = 7/19 = .37

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

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
p(context)	0.16	0.37	0.11	0.26	0.11	

Replacing the counts with joint probabilities, showing the marginals around the outside

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	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
p(context)	0.16	0.37	0.11	0.26	0.11	

- pmi(information,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

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

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

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

- This helps because $P_\alpha(c) > P(c)$ for rare c
- Consider two events, P(a) = .99 and P(b)=.01
- $P_\alpha(a) = \frac{.99^{.75}}{.99^{.75}+.01^{.75}} = .97 \quad P_\alpha(b) = \frac{.01^{.75}}{.01^{.75}+.01^{.75}} = .03$

Use Laplace (add-1) 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) [add-2]

	computer	data	pinch	result	sugar	**p(w)**
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

The Vector Space Model

Measuring similarity: the cosine

Measuring similarity

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

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

  - 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

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

- Dot product is longer 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

The simplest way to modify the dot product to normalize for the vector length is to divide the dot product by the lengths of each of the two vectors. This **normalized dot product** turns out to be the same as the cosine of the angle between the two vectors, following from the definition of the dot product between two vectors $\vec{a}$ and $\vec{b}$:

- 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

Dot product

Unit vectors

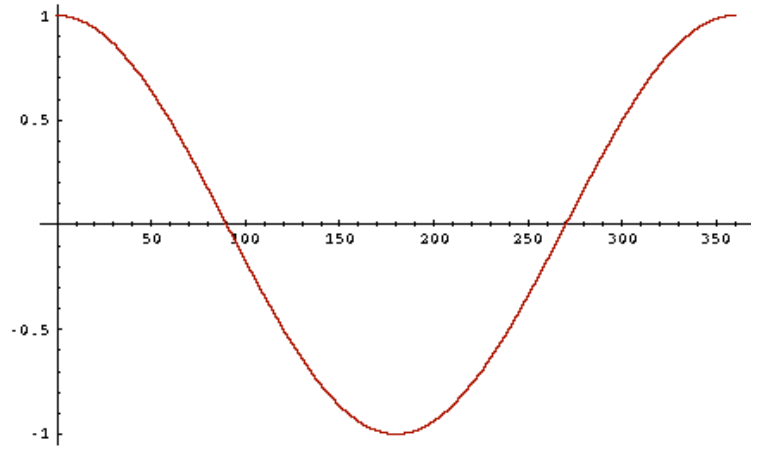
$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

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

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

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

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

Which pair of words is more similar?

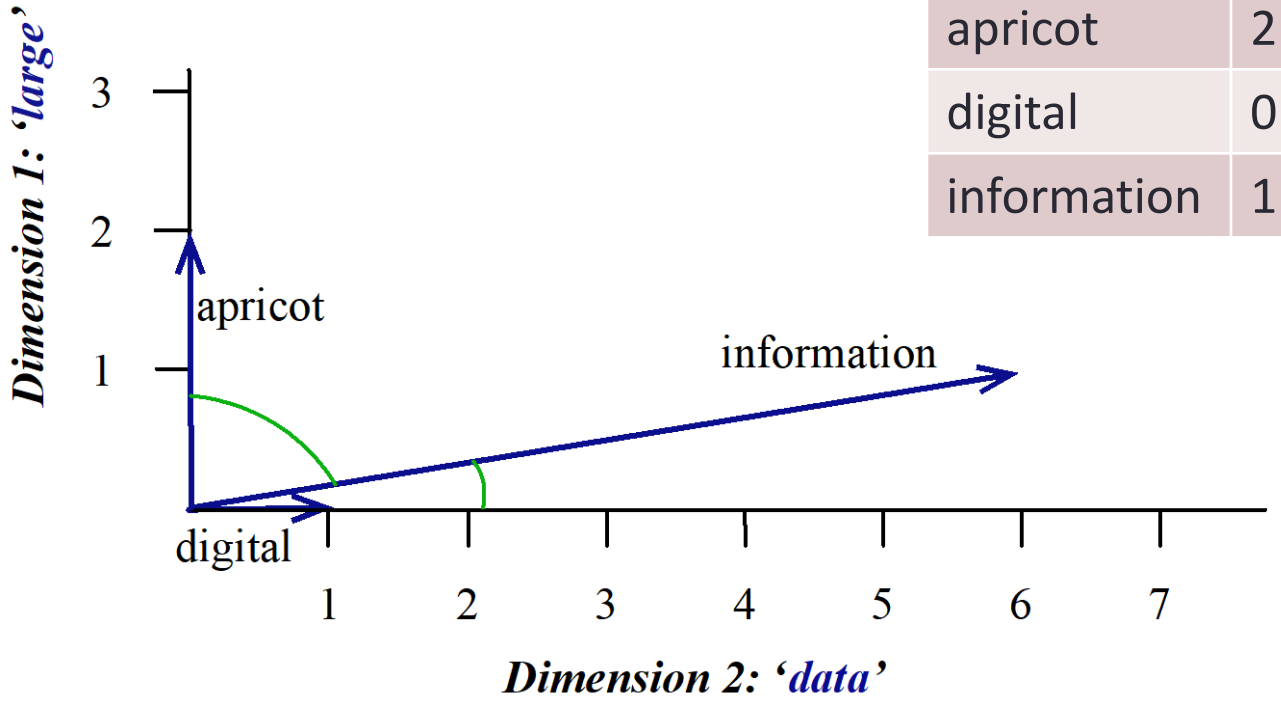
cosine(apricot,information) = $\frac{2+0+0}{\sqrt{2+0+0}\sqrt{1+36+1}} = \frac{2}{\sqrt{2}\sqrt{38}} = .23$

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

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

Visualizing vectors and angles

	large	data
apricot	2	0
digital	0	1
information	1	6



Visualizing Cosine Similarity

The model decides that *information* is closer to *digital* than it is to *apricot*, a result that seems sensible. Fig. 15.10 shows a visualization.

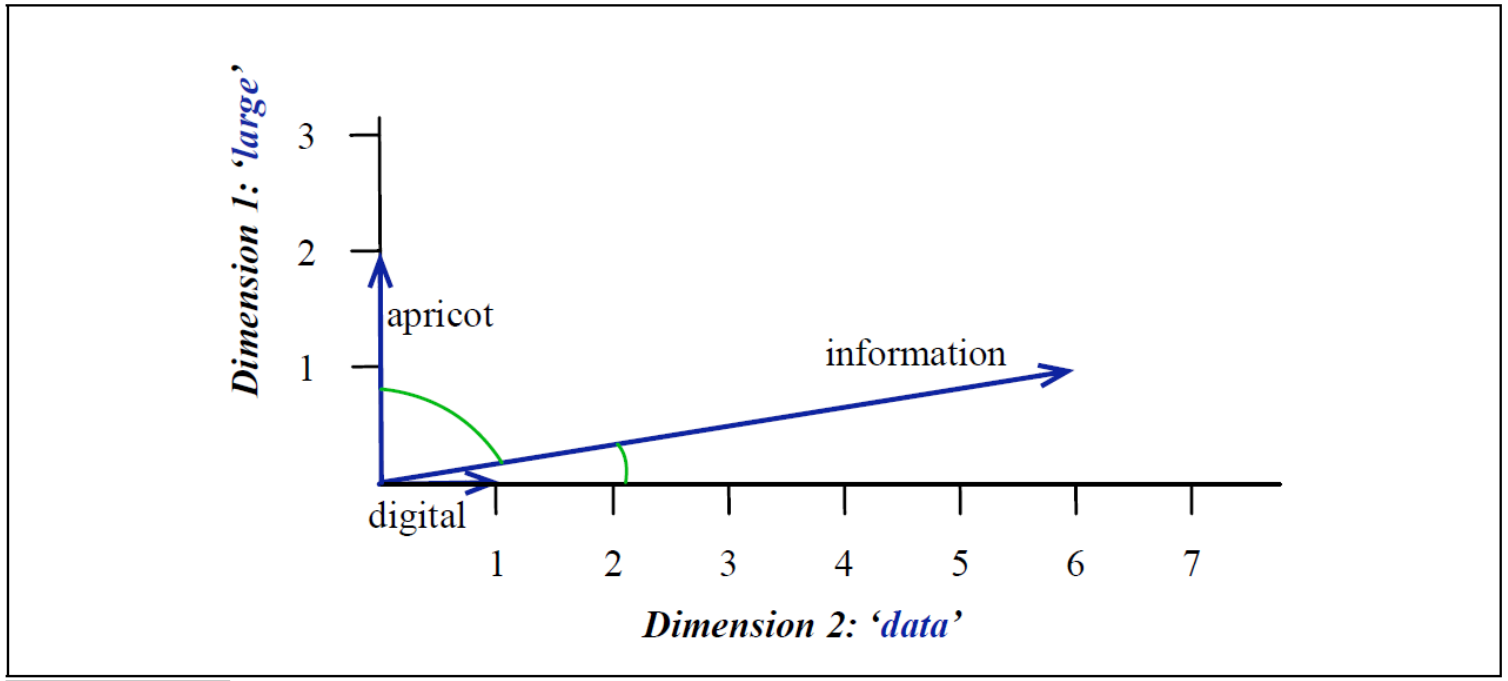
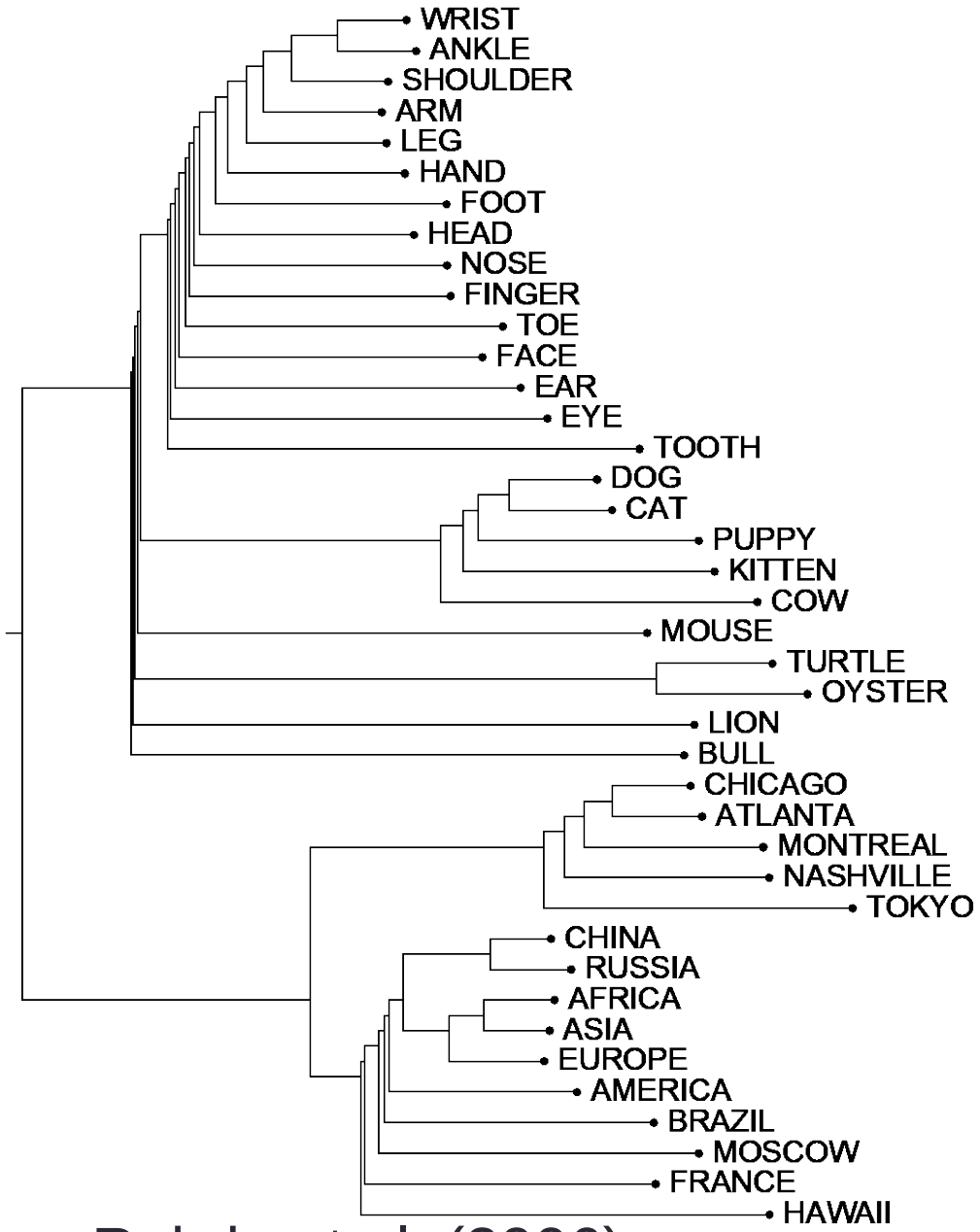


Figure 15.10 A graphical demonstration of the cosine measure of similarity, showing vectors for three words (*apricot*, *digital*, and *information*) in the two dimensional space defined by counts of the words *data* and *large* in the neighborhood. Note that the angle between *digital* and *information* is smaller than the angle between *apricot* and *information*. When two vectors are more similar, the cosine is larger but the angle is smaller; the cosine has its maximum (1) when the angle between two vectors is smallest (0°); the cosine of all other angles is less than 1.

Clustering vectors to visualize similarity in co-occurrence matrices

Example:
Using clustering of vectors as a way to visualize what words are most similar to other ones.



Rohde et al. (2006)

Figure 15.11 Using hierarchical clustering to visualize 4 noun classes from the embeddings produced by Rohde et al. (2006). These embeddings use a window size of ±4, and 14,000 dimensions, with 157 closed-class words removed. Rather than PPMI, these embeddings compute each cell via the positive correlation (the correlation between word pairs, with negative values replaced by zero), followed by a square root. This visualization uses hierarchical clustering, with correlation as the similarity function. From Rohde et al. (2006).

Other possible similarity measures

cosine

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

Jaccard

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

Dice

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

Jensen – Shannon

$$\mathrm{sim}_{\mathrm{JS}}(\vec{v}||\vec{w}) = D\left(\vec{v}\middle|\frac{\vec{v}+\vec{w}}{2}\right) + D\left(\vec{w}\middle|\frac{\vec{v}+\vec{w}}{2}\right)$$

Using syntax to define a word's context

Instead of defining a word's context by nearby words, we could instead define it by the syntactic relations of these neighboring words.

- 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..

Using syntax to define a word's context

Consider the words *duty* and *responsibility*. The similarity between the meanings of these words is mirrored in their syntactic behavior. Both can be modified by adjectives like *additional, administrative, assumed, collective, congressional, constitutional*, and both can be the direct objects of verbs like *assert, assign, assume, attend to, avoid, become, breach* (Lin and Pantel, 2001).

In other words, we could define the dimensions of our context vector not by the presence of a word in a window, but by the presence of a word in a particular dependency (or other grammatical relation), an idea first worked out by Hindle (1990). Since each word can be in a variety of different dependency relations with other words, we'll need to augment the feature space. Each feature is now a pairing of a word and a relation, so instead of a vector of $|V|$ features, we have a vector of $|V| \times R$ features, where R is the number of possible relations.

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

Figure 15.13 Co-occurrence vector for the word *cell*, from Lin (1998), showing grammatical function (dependency) features. Values for each attribute are frequency counts from a 64-million word corpus, parsed by an early version of MINIPAR.

Co-occurrence vectors based on syntactic dependencies

	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

Figure 15.13 Co-occurrence vector for the word *cell*, from Lin (1998), showing grammatical function (dependency) features. Values for each attribute are frequency counts from a 64-million word corpus, parsed by an early version of MINIPAR.

- Since each word can be in a variety of different dependency relations with other words, we'll need to augment the feature space.
- Each feature is now a pairing of a word and a relation, so instead of a vector of |V| features, we have a vector of |V|x R features, where R is the number of possible relations.
- The figure shows a schematic early example of such a vector, showing one row for the word cell.
- As the value of each attribute we have shown the raw frequency of the feature co-occurring with cell.

Syntactic dependencies for dimensions

- Alternative (Padó and Lapata 2007):
  - Instead of having a |V| x R|V| matrix
  - Have a |V| x |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("cell","absorb") = count(subj(cell,absorb)) + count(obj(cell,absorb)) + count(pobj(cell,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*

$$\text{idf}_i = \log\left(\frac{N}{df_i}\right)$$

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

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

The Vector Space Model

Evaluating Similarity

Evaluating similarity

- Extrinsic (task-based, end-to-end) Evaluation:
  - The most important evaluation metric for vector models is extrinsic evaluation on tasks; adding them as features into any NLP task and seeing whether this improves performance.
    - ✓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`

Evaluating similarity

Intrinsic Evaluation:

- The most common metric is to test their performance on similarity, and in particular on computing the correlation between an algorithm's word similarity scores and word similarity ratings assigned by humans.
- The various sets of human judgments are the same as for thesaurus-based similarity, summarized here for convenience.
  - WordSim-353 (Finkelstein et al., 2002) is a commonly used set of of ratings from 0 to 10 for 353 noun pairs; for example (plane, car) had an average score of 5.77.
  - SimLex-999 (Hill et al., 2015) is a more difficult dataset that quantifies similarity (cup, mug) rather than relatedness (cup, coffee), and including both concrete and abstract adjective, noun and verb pairs.
  - The TOEFL dataset is a set of 80 questions, each consisting of a target word with 4 additional word choices; the task is to choose which is the correct synonym, as in the example:
    - ✓ Levied is closest in meaning to: imposed, believed, requested, correlated.
- All of these datasets present words without context.

Evaluating similarity

Intrinsic Evaluation:

- Slightly more realistic are intrinsic similarity tasks that include context.
  - ✓ The Stanford Contextual Word Similarity (SCWS) dataset offers a richer evaluation scenario, giving human judgments on 2,003 pairs of words in their sentential context, including nouns, verbs, and adjectives.
  - ✓ This dataset enables the evaluation of word similarity algorithms that can make use of context words.
- The semantic textual similarity task evaluates the performance of sentence-level similarity algorithms, consisting of a set of pairs of sentences, each pair with human-labeled similarity scores.

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
    - Brown clusters 5-20 binary dimensions.