

## CS 6120/CS4120: Natural Language Processing

Instructor: Prof. Lu Wang  
 College of Computer and Information Science  
 Northeastern University  
 Webpage: [www.ccs.neu.edu/home/luwang](http://www.ccs.neu.edu/home/luwang)

## Outline

- Vector Semantics
- Sparse representation
  - Pointwise Mutual Information (PMI)
- Dense representation
  - Singular Value Decomposition (SVD)
  - Neural Language Model (Word2Vec) (next lecture)

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”

- Automatically constructed clusters of semantically similar words (Charniak, 1997):

Friday	Monday	Thursday	Wednesday	Tuesday	Saturday	Sunday
People	guys	folks	fellows	CEOs	commies	blocks
water	gas	cola	liquid	acid	carbon	steam
shale						
that	the	theat				
head	body	hands	eyes	voice	arm	seat
eye	hair	mouth				

## Smoothing for statistical language models

- Two alternative guesses of speech recognizer:

For breakfast, she ate durian.

For breakfast, she ate Dorian.

- Our corpus contains neither “ate durian” nor “ate Dorian”
- But, our corpus contains “ate orange”, “ate banana”

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

- Example:
  - What is *tesguino*?

### Intuition of distributional word similarity

- Example:
  - A bottle of *tesguino* is on the table
  - Everybody likes *tesguino*
  - Tesguino* makes you drunk
  - We make *tesguino* out of corn.
- From context words humans can guess *tesguino* means
  - an alcoholic beverage like **beer**
- Intuition for algorithm:
  - Two words are similar if they have similar word contexts.

### Four kinds of vector models

#### Sparse vector representations

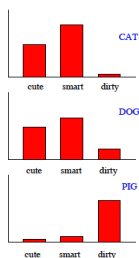
1. Mutual-information weighted word co-occurrence matrices

#### Dense vector representations:

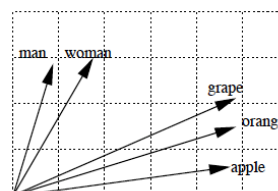
2. Singular value decomposition (and Latent Semantic Analysis)
3. Neural-network-inspired models (skip-grams, CBOW)
4. Brown clusters

### 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**”



### Sample Lexical Vector Space



Term-context matrix for word similarity

Term-document matrix

- Each cell: count of term (or word)  $t$  in 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

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-word matrix for word similarity

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

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,

	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 matrix

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of  $\pm k$  (e.g.  $k=4$ ) 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 **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

## Sample Word-Word matrix

	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.

## 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
    - $\pm 1-3$  very syntactic
    - You may see playing is similar to cooking or singing, played is similar to cooked or sang
  - The longer the windows, the more **semantic** the representation
    - $\pm 4-10$  more semantic

## 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

### 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?

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

### Pointwise Mutual Information

**Pointwise mutual information:**

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

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

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

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

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

### 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 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"

### 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:

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

### Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts, e.g. in the form of words)

•  $f_{ij}$  is number of times  $w_i$  occurs in context  $c_j$

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

	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

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

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

		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

$p(w=\text{information}, c=\text{data}) = \frac{6}{19} = .32$   
 $p(w=\text{information}) = \frac{11}{19} = .58$   
 $p(c=\text{data}) = \frac{7}{19} = .37$

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

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

		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	

$$pmi_{ij} = \log_2 \frac{P_{ij}}{P_i \cdot P_j}$$

		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
<b>p(context)</b>		0.16	0.37	0.11	0.26	0.11	

$\bullet pmi(\text{information}, \text{data}) = \log_2 ( .32 / (.37 * .58) ) = .57$

		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-k smoothing (which has a similar effect)

- ### Weighting PMI: Giving rare context words slightly higher probability
- Raise the context probabilities to  $\alpha = 0.75$ :
 
$$PPMI_{\alpha}(w, c) = \max(\log_2 \frac{P(w, c)}{P(w)P_{\alpha}(c)}, 0)$$

$$P_{\alpha}(c) = \frac{\text{count}(c)^{\alpha}}{\sum_c \text{count}(c)^{\alpha}}$$
  - This helps because  $P_{\alpha}(c) > P(c)$  for rare c
  - Consider two events,  $P(a) = .99$  and  $P(b) = .01$  (here we use probability to show the effect)
  - $P_{\alpha}(a) = \frac{.99^{.75}}{.99^{.75} + .01^{.75}} = .97$   $P_{\alpha}(b) = \frac{.01^{.75}}{.99^{.75} + .01^{.75}} = .03$

### Add-k smoothing

		Add-2 Smoothed Count				
		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]					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
<b>p(context)</b>		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

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

- 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

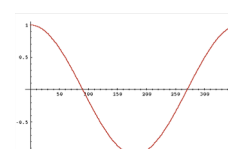
$$\cos(\vec{v}, \vec{w}) = \frac{\vec{v} \cdot \vec{w}}{|\vec{v}| |\vec{w}|} = \frac{\sum_{i=1}^N v_i w_i}{\sqrt{\sum_{i=1}^N v_i^2} \sqrt{\sum_{i=1}^N w_i^2}}$$

$v_i$  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

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

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

Which pair of words is more similar?

$$\text{cosine}(\text{apricot}, \text{information}) = \frac{2+0+0}{\sqrt{4+0+0} \sqrt{1+36+1}} = \frac{2}{\sqrt{4} \sqrt{38}} = .23$$

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

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

Visualizing vectors and angles

	large	data
apricot	2	0
digital	0	1
information	1	6

Clustering vectors to visualize similarity in co-occurrence matrices

Rohde et al. (2006)

Other possible similarity measures

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

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

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

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* :

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



Co-occurrence

subj-of, absorb
subj-of, adapt
subj-of, behave
:
obj-of, inside
obj-of, into
:
nmod-of, abnormality
nmod-of, anemia
nmod-of, architecture
:
obj-of, attack
obj-of, call

syntactic dependencies

Retrieval and Clustering of Similar Words\*

Grammatical relations

R/V/

- Each dimension: a context
  - Subject-of- "absorb"
- Instead of a vector of /V/
- Example: counts for the v

### 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  $M("cell", "absorb") = \text{count}(\text{subj}(\text{cell}, \text{absorb})) + \text{count}(\text{obj}(\text{cell}, \text{absorb})) + \text{count}(\text{obj}(\text{cell}, \text{absorb}))$ , 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

- The combination of two factors
  - Term frequency** (Luhn 1957): frequency of the word (can be logged)
  - Inverse document frequency** (IDF) (Spark Jones 1972)
    - N is the total number of documents
    - $df_i$  = "document frequency of word i"
    - = number of documents with word i

$$w_{ij} = tf_{ij} \cdot idf_i \quad \Rightarrow \quad idf_i = \log \left( \frac{N}{df_i} \right)$$

### 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

### Evaluating similarity (Revisit)

- 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.  $\text{sim}(\text{plane}, \text{car}) = 5.77$
  - Taking TOEFL multiple-choice vocabulary tests
    - Leviad** is closest in meaning to:
      - imposed, believed, requested, correlated

Summary and next step

- **Distributional (vector) models of meaning**
  - **Sparse** (PPMI-weighted word-word co-occurrence matrices)
  - **Dense:**
    - Word-word SVD (50-2000 dimensions)
    - Skip-grams and CBOW (100-1000 dimensions)