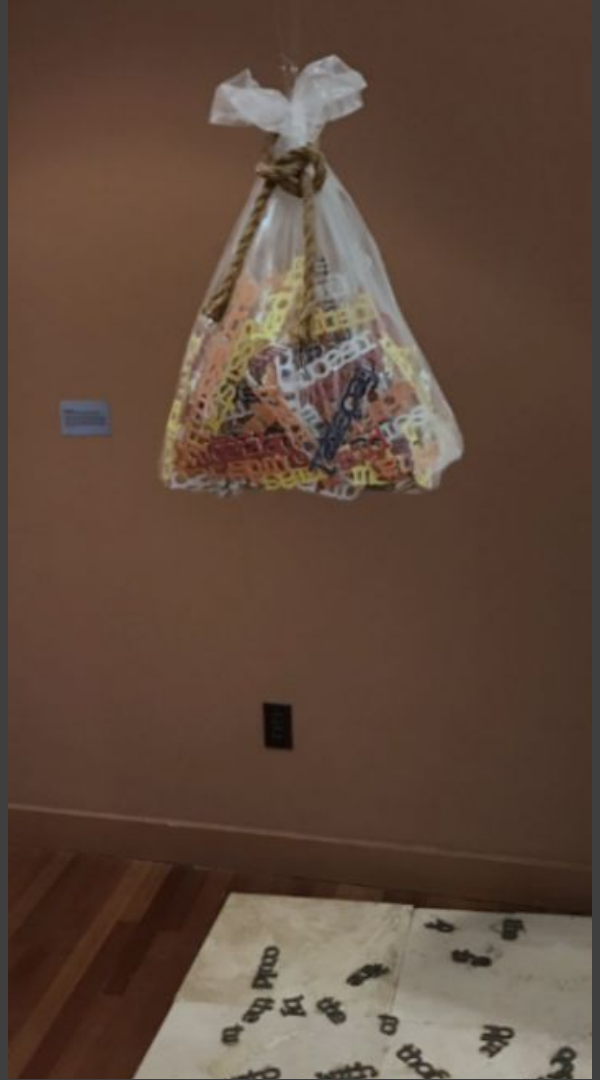


CS 2731 Introduction to Natural Language Processing

Session 3: Bag of words, tf-idf, PPMI

Michael Miller Yoder

September 6, 2023

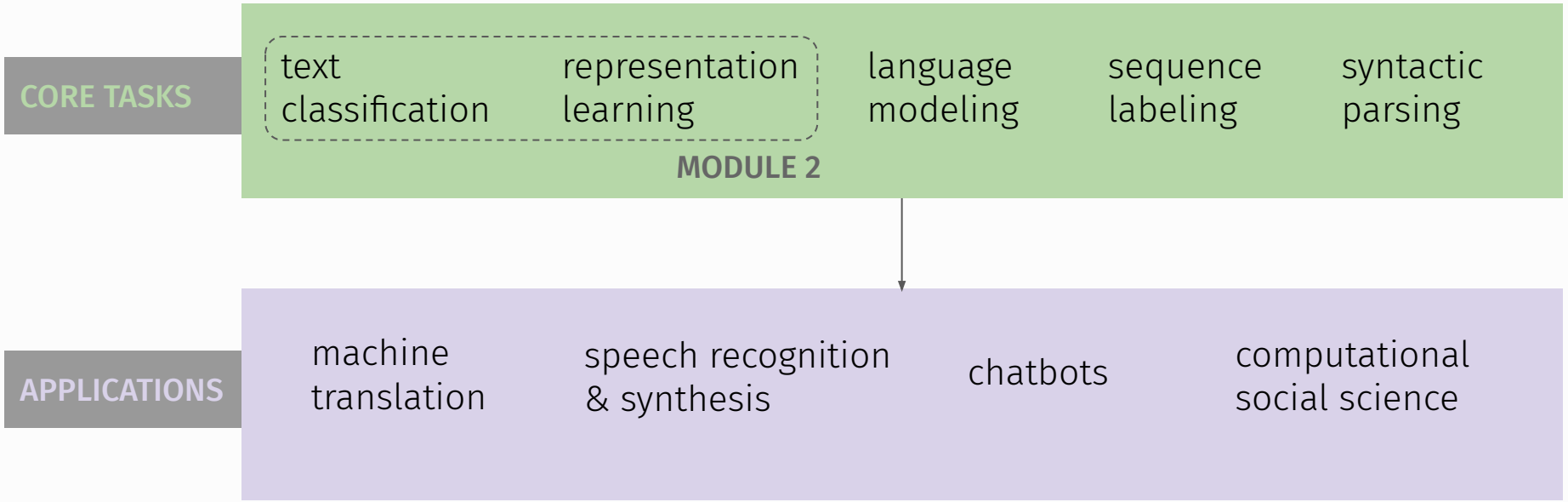
School of Computing and Information

- Course logistics
- Term-document and term-term matrices
- Cosine similarity
- Tf-idf weighting
- PPMI weighting

Course logistics

- Reading quizzes due noon before class
- Homework 1
  - Will be released today (check website or Canvas)
  - Programming problems, write report based on results
  - Due next Thu, Sep 13 at 11:59pm
- Course project matching
  - By Mon, Sep 11
  - We will email you

Core tasks and applications of NLP



Text classification

"My dear Mr. Bennet," said his lady to him one day, "have you heard that Netherfield Park is let at last?"

ROMANCE

Pride and Prejudice

DIALOG

Bag of words document representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Slide from Jurafsky & Martin

Documents Can Be Represented as Bags of Words

A BAG OF WORDS is a BAG or MULTISET of the words in a document.

- Like a set, except that identical elements can appear multiple times
- You could also think of it like a `Counter` object in Python

```
bow = {'and': 23502,
       'or': 12342',
       'the': 54939508,
       ...
       'hippopotamus': 1}
```

- If you take out sequencing information, any document can be viewed as a bag of words.

Slide credit: David Mortensen

Bags of Words Can Be Represented as Sparse Vectors

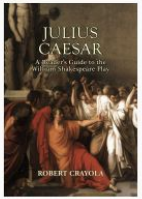
- So far, we've represented bags of words as dense MAPS, but sometimes it is useful to represent them as sparse vectors
- Led V be the sent of all words in the document collection D. Then our BoW vectors for documents in D will have $|V|$ dimensions.
- Each dimension corresponds to a word `type` and the value at this dimensions corresponds to the number of TOKENS of that type in the document that the vector is representing

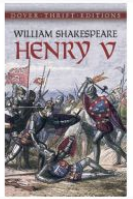
Slide credit: David Mortensen

Term-document and term-term matrices

Term-document matrix

- Each cell is the count of term t in a document d ($tf_{t,d}$).
- Each document is a count vector in $\mathbb{N}^V$, a column below.



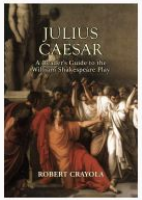


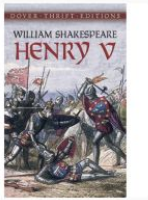
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Slide credit: David Mortensen

Term-document matrix

- Two documents are similar of their vectors are similar.

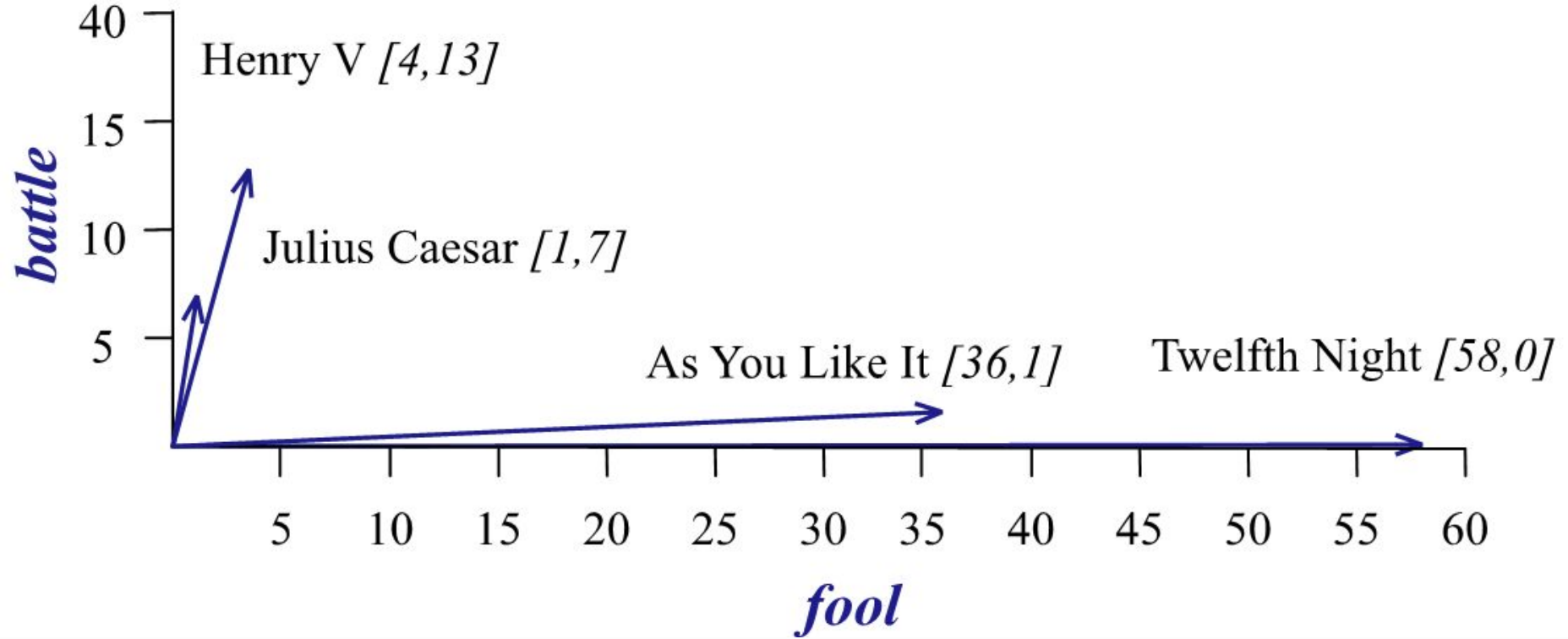




	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Slide credit: David Mortensen

Visualizing document vectors



Slide from Jurafsky & Martin

Vectors are the basis of information retrieval

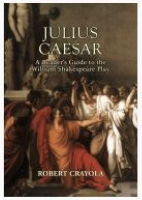
	As You Like It	*Twelfth Night*	*Julius Caesar*	*Henry V*
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

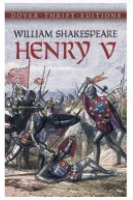
- Vectors for comedies are different from tragedies
- Comedies have more *fools* and fewer *battles*

Slide adapted from Jurafsky & Martin

Term-document matrix: word vectors

Two words are similar if their vectors are similar.





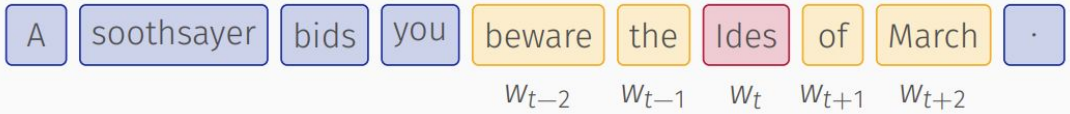
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

- *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"

Slide adapted from David Mortensen, Jurafsky & Martin

Term-term matrix (or word-word or word-context matrix)

- Instead of entire documents, use smaller contexts
  - Paragraph
  - Window of a few words (e.g. 3, 5, 7):



- A word is now defined by a vector over counts of words in context.
  - If a word w_j occurs in the context of w_i, increase $count_{ij}$.
- Assuming we have V words,
  - Each vector is now of length V.
  - The word-word matrix is $V \times V$.

Slide credit: David Mortensen

Sample Contexts of ±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

	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
⋮						

Slide credit: David Mortensen

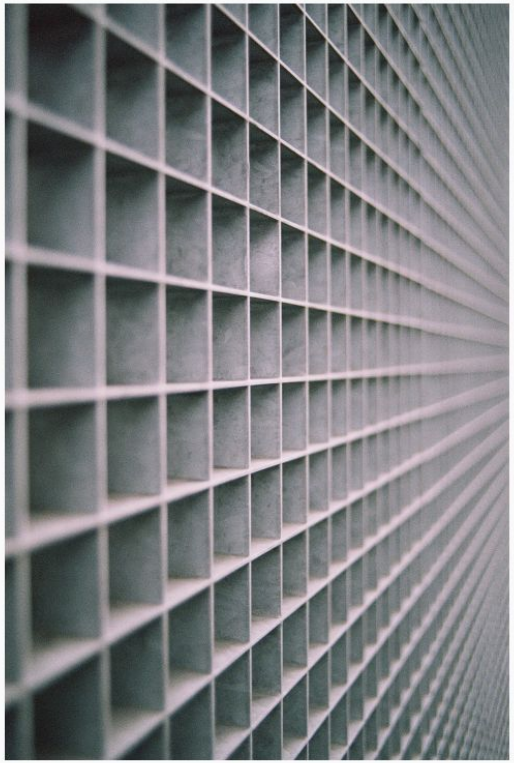
The Word–Word Matrix

We showed only a 4×6 matrix, but the real matrix is $50,000 \times 50,000$.

- So it is 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 the goals:

- The **smaller** the context ($\pm 1 - 3$) , the more **syntactic** the representation
- The **larger** the context ($\pm 4 - 10$), the more **semantic** the representation



Slide credit: David Mortensen

Types of Co-occurence between Two Words

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):

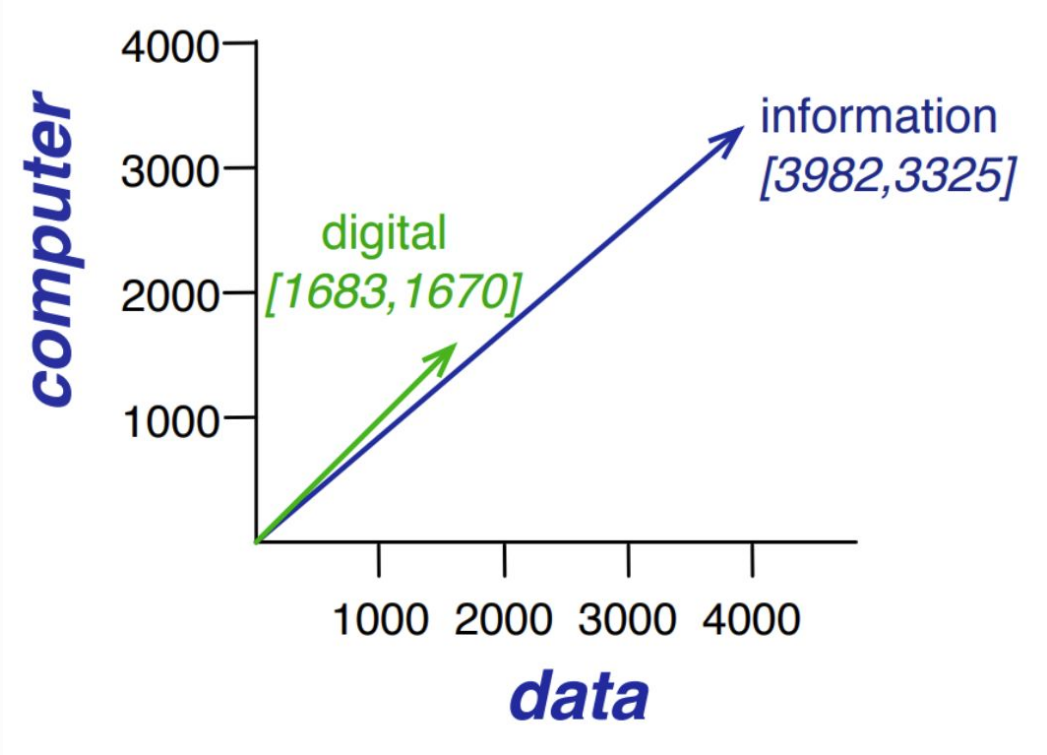
- They have similar neighbors.
- *wrote* is a second-order associate of words like *said* or *remarked*.

Slide credit: David Mortensen

Cosine similarity

Measuring similarity between word or document vectors

- Do we care about magnitude/word frequencies? (No)



Slide adapted from Jurafsky & Martin

Cosine is Used to Measure the Similarity between Word Vectors

- Given two target words represented with vectors $\boldsymbol{v}$ and $\boldsymbol{w}$.
- The **dot product** or **inner product** is usually used as the basis for similarity.

$$\boldsymbol{v} \cdot \boldsymbol{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \cdots + v_N w_N = |\boldsymbol{v}||\boldsymbol{w}| \cos \theta$$

- $\boldsymbol{v} \cdot \boldsymbol{w}$ is high when two vectors have large values in the same dimensions.
- $\boldsymbol{v} \cdot \boldsymbol{w}$ is low (in fact 0) with zeros in complementary distribution.
- We also do not want the similarity to be sensitive to word-frequency.
- So normalize by vector length and use the cosine as the similarity

$$\cos(V, W) = \frac{V \cdot W}{|V||W|}$$

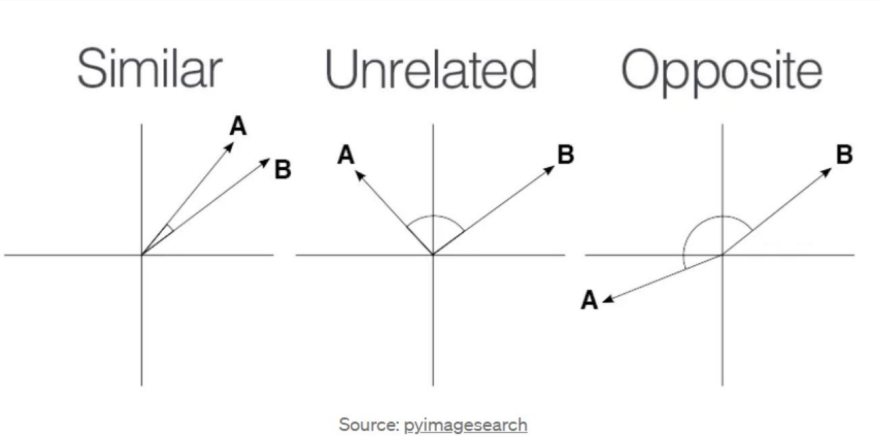
Slide credit: David Mortensen

Cosine as a similarity metric for vectors

-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

Slide adapted from Jurafsky & Martin, David Mortensen

Tf-idf weighting

Raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
  - Whether in term-document or term-term matrices
- 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
  - 2 documents that use a lot of *the* are not necessarily similar
- It's a paradox! How can we balance these two conflicting constraints?

Slide adapted from Jurafsky & Martin

Weighting words in term-document and term-term matrices

- **tf-idf** (term frequency-inverse document frequency)
  - For representing documents with their most unique words (for text classification, information retrieval)
  - Term-document matrix
- **PPMI** (positive pointwise mutual information)
  - For finding associations between words (which appear more often together than chance?)
  - Term-term matrix

Slide adapted from Jurafsky & Martin

Term frequency (tf)

$tf_{t,d} = count(t,d)$

Instead of using raw count, we squash a bit:

$tf_{t,d} = \log_{10}(count(t,d)+1)$

Slide adapted from Jurafsky & Martin

Document frequency (df)

df_t is the number of documents term *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

Slide adapted from Jurafsky & Martin

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
- Documents can be whatever you want! (Full documents, paragraphs, etc)

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

Slide adapted from Jurafsky & Martin

tf-idf Controls for Frequent but Uninformative Words

Some words are very common in a given document because they are common across all documents (e.g., *the*). They are not discriminative. `tf-idf` (product of term frequency and inverse document frequency) addresses this:

$$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d) + 1)$$

$$\text{idf}_f = \log_{10} \frac{N}{\text{df}_t}$$

$$\text{tf-idf}(t,d) = \text{tf}_{t,d} \cdot \text{idf}_t$$

Slide adapted from Jurafsky & Martin, David Mortensen

Final tf-idf weighted values

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

Positive pointwise mutual information (PPMI)

The Problem with Raw Counts

The **to** in **to walk** doesn't tell us as much about **walk** as the **slowly** in **walk slowly**.

Slide credit: David Mortensen

Problem with Raw Counts

- Raw word frequency is not a great measure of association between words.
- It is very skewed: “the” and “of” are very frequent, but maybe not the most discriminative.
- We would rather have a measure that asks whether a context word is **particularly informative** about the target word.

Positive Pointwise Mutual Information (PPMI)

Slide credit: David Mortensen

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 2 words [Church+Hanks 1989]
  - Do words *x* and *y* co-occur more than if they were independent?

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

Slide adapted from Jurafsky & Martin

- In computational linguistics, PMI has been used for finding collocations and associations between words.

word 1	*word 2*	*count word 1*	*count word 2*	count of co-occurrences	PMI
puerto	rico	1938	1311	1159	10.0349081703
hong	kong	2438	2694	2205	9.72831972408
los	angeles	3501	2808	2791	9.56067615065
carbon	dioxide	4265	1353	1032	9.09852946116
prize	laureate	5131	1676	1210	8.85870710982
san	francisco	5237	2477	1779	8.83305176711
nobel	prize	4098	5131	2498	8.68948811416
ice	hockey	5607	3002	1933	8.6555759741
star	trek	8264	1594	1489	8.63974676575
car	driver	5578	2749	1384	8.41470768304
it	the	283891	3293296	3347	-1.72037278119
are	of	234458	1761436	1019	-2.09254205335
this	the	199882	3293296	1211	-2.38612756961
is	of	565679	1761436	1562	-2.54614706831
and	of	1375396	1761436	2949	-2.79911817902
a	and	984442	1375396	1457	-2.92239510038
in	and	1187652	1375396	1537	-3.05660070757
to	and	1025659	1375396	1286	-3.08825363041
to	in	1025659	1187652	1066	-3.12911348956
of	and	1761436	1375396	1190	-3.70663100173

Slide credit: David Mortensen

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}.
  - Furthermore it's not clear people are good at "unrelatedness".
- So we just replace negative PMI values by 0.

$$\text{PPMI}(w, c) = max\left(\log_2 \frac{p(w, c)}{p(w)p(c)}, 0\right)$$

Slide credit: David Mortensen

Computing PPMI on a Term-Context Matrix

- We have matrix F with V rows (words) and C columns (contexts) (in general $C = V$)
- f_{ij} is how many times word w_i co-occurs in the context of the word c_j.

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{V}(\sum_{j=1}^{C} f_{ij})}$$

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

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \max(pmi_{ij}, 0)$$

Slide credit: David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part I)

	computer	data	pinch	result	sugar	
apricot	0	0	1	0	1	2
pineapple	0	0	1	0	1	2
digital	2	1	0	1	0	4
information	1	6	0	4	0	11
	3	7	2	5	2	19

$$p(w = information, c = data) = \frac{6}{19} = 0.32$$

$$p(w = information) = \frac{11}{19} = 0.58 \quad p(c = data) = \frac{7}{19} = 0.37$$

$$pmi(information, data) = \log_2 \frac{0.32}{0.37 \cdot 0.58} \approx 0.58$$

Slide credit: David Mortensen

Worked Example: Computing PPMI from Term-Context Matrix (Part II)

$PPMI(w, c)$

	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	0.32	-	0.47	-

Slide credit: David Mortensen

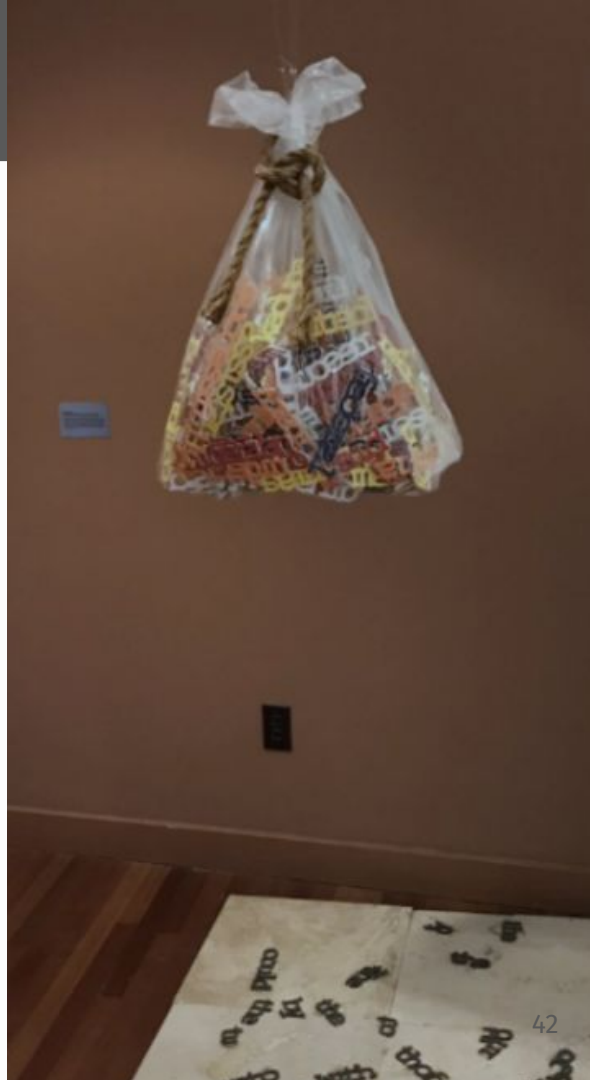
Issues with PPMI

- 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)

Slide credit: David Mortensen

Conclusion

- **Bag of words** representations of documents
  - Just counts of terms in documents (no order info)
  - Can be represented in vector form as...
- **Term-document matrices**
- **Term-term (word-word) matrices**
  - How many times words are used in contexts of other words
- **Cosine** to measure similarity between vectors for documents or words
- Downweighting words that appear frequently
  - **tf-idf** for document representations
    - Downweight terms that appear across many documents
  - **PPMI** for word associations
    - Downweight words that appear with many other words



Questions?

Homework 1 released today,
due next Thu, Sep 13