TTIC 31190: Natural Language Processing

Kevin Gimpel

Spring 2018

Lecture 2: Words, Morphology, Distributional Word Vectors

- All materials are posted on the course website:

`ttic.uchicago.edu/~kgimpel/teaching/31190-s18/index.html`

- Assignment 1 has been posted
- Due 6:00 pm on Wednesday, April 11th

- My office hours are Mondays 3-4pm, #531 (or by appointment)
- TA office hours are Wednesdays 3-4pm, #501

Roadmap

- words, morphology, lexical semantics
- text classification
- simple neural methods for NLP
- language modeling and word embeddings
- recurrent/recursive/convolutional networks in NLP
- sequence labeling, HMMs, dynamic programming
- syntax and syntactic parsing
- semantics, compositionality, semantic parsing
- machine translation and other NLP tasks

Words

- types and tokens
- morphology
- distributional word vectors
- word sense and lexical semantics

Types and Tokens

- once text has been tokenized, let's count the words
- types: entries in the vocabulary
- tokens: instances of types in a corpus
- example sentence: *If they want to go , they should go .*
  - how many types? 8
  - how many tokens? 10
- type/token ratio: useful statistic of a corpus (here, 0.8)

Higher Type/Token Ratio?

- Wikipedia vs Simple English Wikipedia?
  - Wikipedia
- Wikipedia vs Newswire?
  - Wikipedia
- Wikipedia vs Tweets?
  - Tweets (once you have 1 million or more tokens)

“really” on Twitter

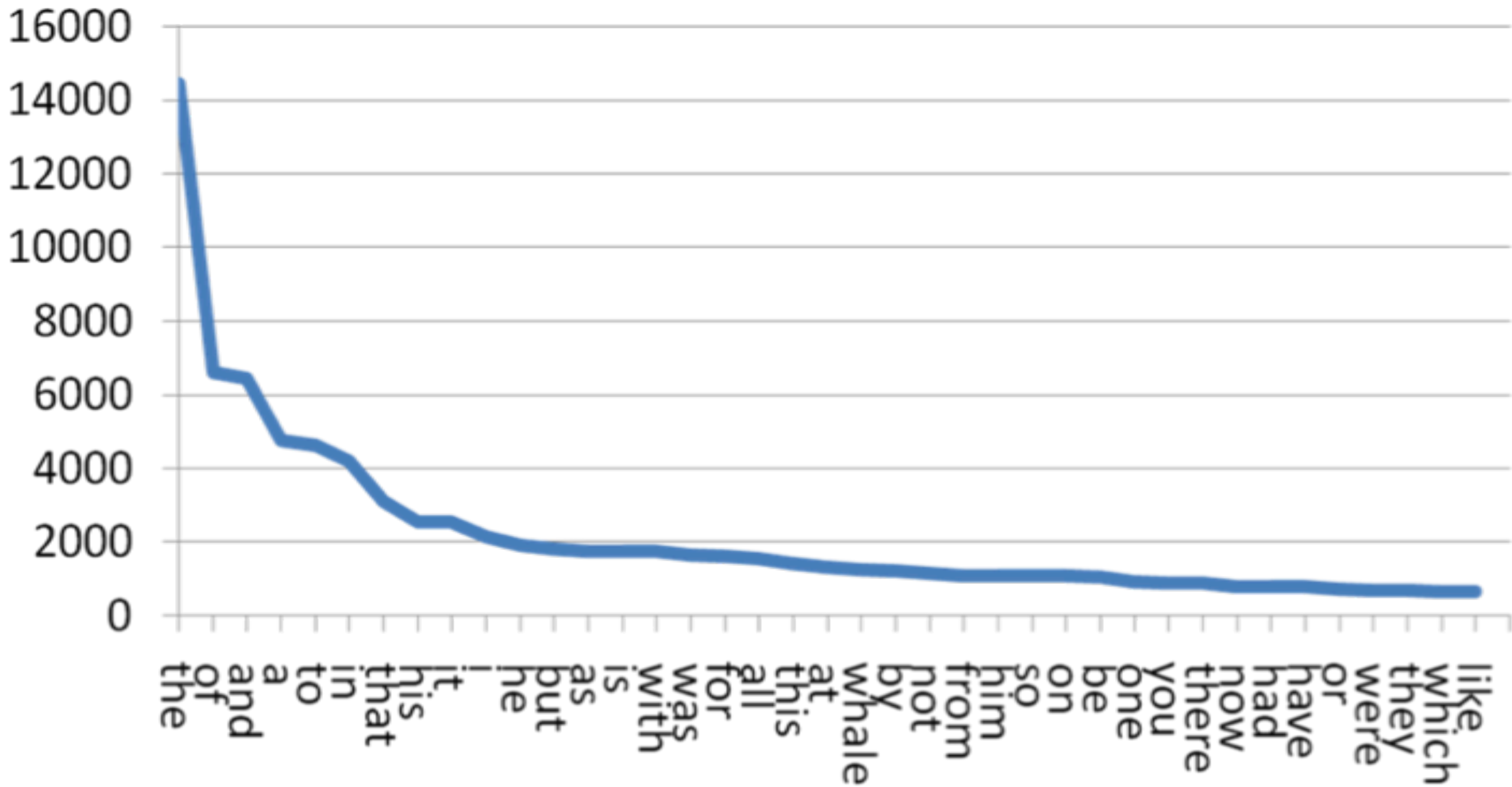
224571	really
1189	rly
1119	realy
731	rlly
590	reallly
234	realllly
216	reallyy
156	relly
146	reallllly
132	rily
104	reallyyy
89	reeeally
89	realllllly
84	reaaally
82	reaally
72	reeeeally
65	reaaaally
57	reallyyyy
53	rilly
50	reallllllly
48	reeeeeally
41	reeally
38	really2
37	reaaaaally
35	reallyyyyy
31	reely
30	realllyyy
27	realllyy
27	reaaly
26	realllyyyy
25	realllllllly
22	reaaallly
21	really-
19	reeaally
18	reallllyyy
16	reaaaallly
15	realyy
15	reallyreally
15	reallllyy
15	reallllllllly
15	reaallly
14	reeeeeeally
14	reallllyyyy
13	reeeaaally
12	rreally
12	reaaaaaally
11	reeeeallly
11	reeeallly
11	reallllllyyy
11	reaallyy
10	reallyreallyreally
10	reaaaly
9	reeeeeeeally
9	reallys
9	really-really
9	r)eally
8	reeeaally

How many words are there?

- a bit surprising: vocabulary continues to grow in any actual dataset
- you'll just never see all the words

How are words distributed?

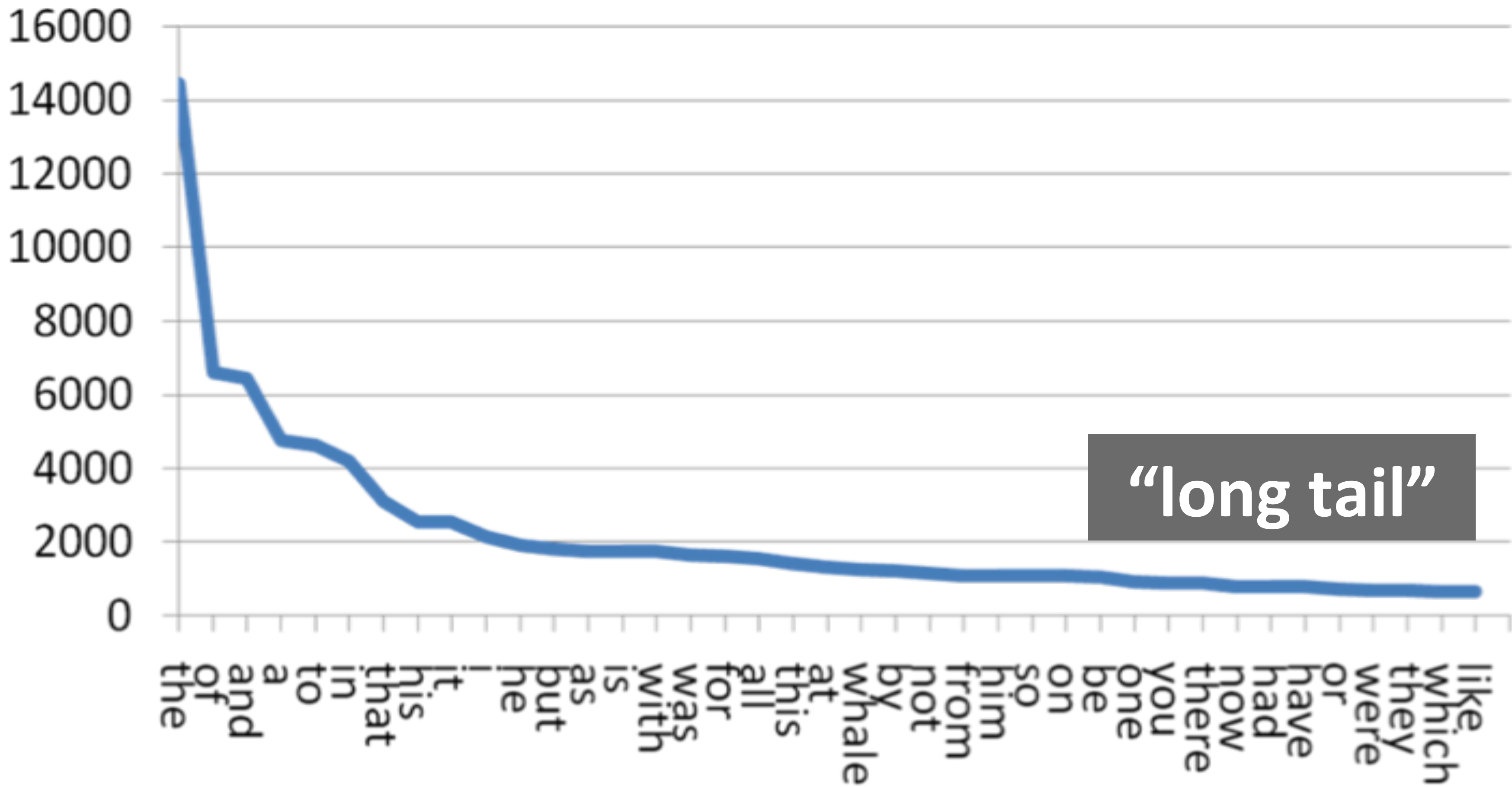
- Zipf's law: frequency of a word is inversely proportional to its rank



blog EMIS

How are words distributed?

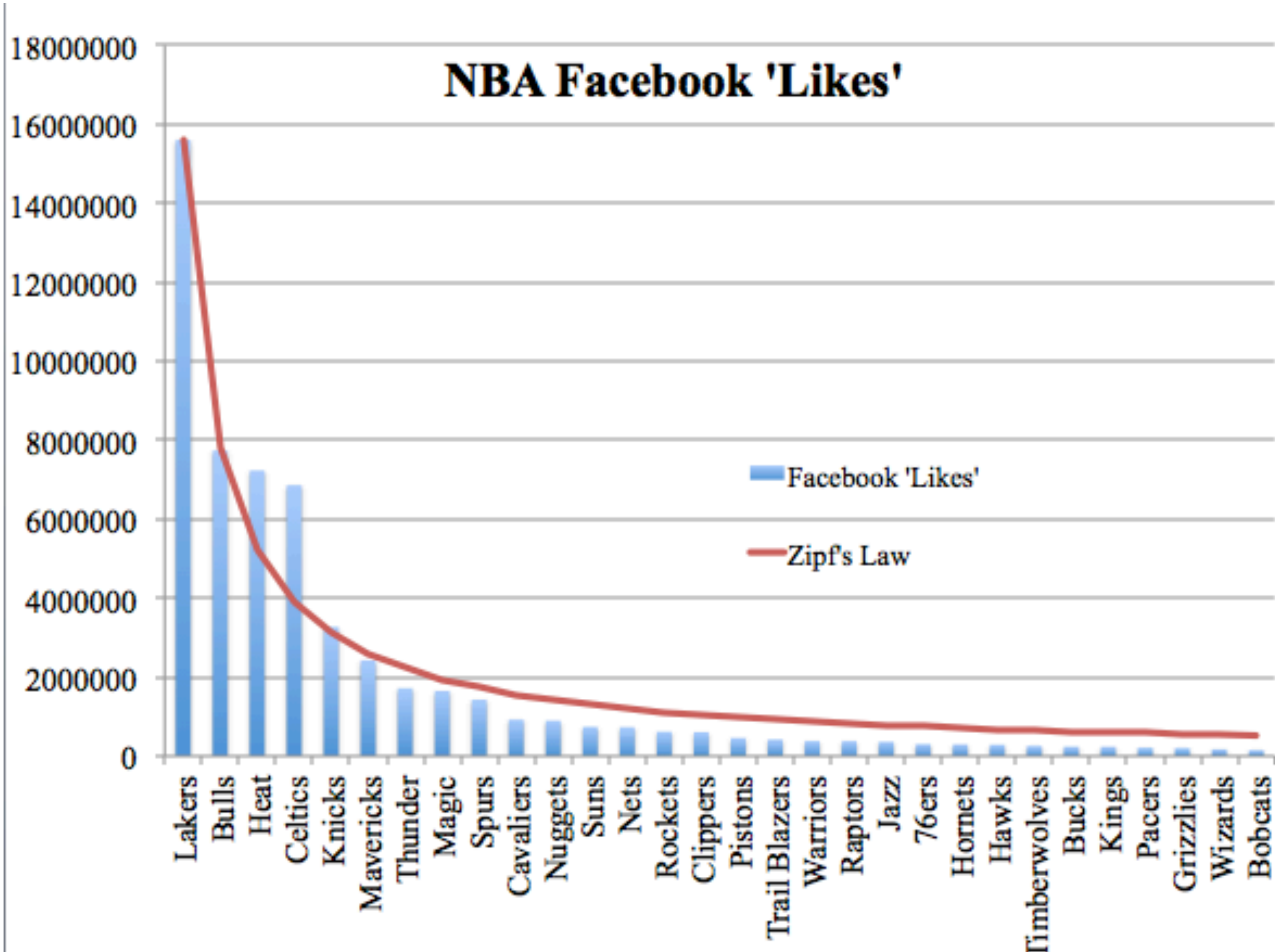
- Zipf's law: frequency of a word is inversely proportional to its rank



blog EMIS

Zipf's Law

- also predicts other kinds of data: population of cities in a country, revenue of different companies, etc.



The Laurentian University Sports Analytics Group

The Long Tail

- there are so many word types!
- but words have **internal structure**

Words

- types and tokens
- morphology
- distributional word vectors
- word sense and lexical semantics

Type/Token Ratio across Languages

- high type/token ratio →
- low type/token ratio →

Type/Token Ratio across Languages

- high type/token ratio → rich morphology
- low type/token ratio → poor morphology

Morphology

- morphemes:
  - the small meaningful units that make up words
  - stems: core meaning-bearing units
  - affixes: bits and pieces that adhere to stems
    - often with grammatical functions

Kinds of Word Formation

- inflection: modifying a word with an affix to change its grammatical function (tense, number, etc.)
  - result is a "different form of the same word"
  - examples: *book* → *books*, *walk* → *walked*

Kinds of Word Formation

- inflection: modifying a word with an affix to change its grammatical function (tense, number, etc.)
  - result is a "different form of the same word"
  - examples: *book* → *books*, *walk* → *walked*
- derivation: adding an affix to a stem to create a new word
  - examples: *great* → *greatly*, *great* → *greatness*

Kinds of Word Formation

- inflection: modifying a word with an affix to change its grammatical function (tense, number, etc.)
  - result is a "different form of the same word"
  - examples: *book* → *books*, *walk* → *walked*
- derivation: adding an affix to a stem to create a new word
  - examples: *great* → *greatly*, *great* → *greatness*
- compounding: combining two stems
  - examples: *lawsuit*, *keyboard*, *bookcase*

Morphology

- usually, morphological derivation is simply splitting a word into its morphemes:
  – walked = walk + ed
  – greatness = great + ness
- but it can actually be a hierarchical structure
  – unbreakable = ?

Morphology

- usually, morphological derivation is simply splitting a word into its morphemes:
  - walked = walk + ed
  - greatness = great + ness
- but it can actually be a hierarchical structure
  - unbreakable = un + (break + able)

Morphology

- ambiguity in hierarchical morphological decomposition?
  – rare, but it does happen

Morphology

- ambiguity in hierarchical morphological decomposition?
  - rare, but it does happen
  - unlockable = un + lock + able
    - what does this word mean?

Morphology

- ambiguity in hierarchical morphological decomposition?
  - rare, but it does happen
  - unlockable = un + lock + able
    - what does this word mean?
    - (un+lock)+able: “able to be unlocked”
    - un+(lock+able): “unable to be locked”

Morphology in NLP

- two common tasks:
  - lemmatization
  - stemming

Lemmatization

- lemmatization: reduce inflections or variant forms to base form
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- have to find correct dictionary headword form
- e.g., for machine translation:
  - Spanish *quiero* (‘I want’), *quieres* (‘you want’) same lemma as *querer* ‘want’

J&M/SLP3

Stemming

- stemming: reduces words to their stems via crude chopping of affixes
  - e.g., *automate(s), automatic, automation* all reduced to *automat*
  - language dependent
  - key step in information retrieval

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

J&M/SLP3

Porter's algorithm
The most common English stemmer

Step 1a

```
sses → ss    caresses → caress
ies  → i     ponies   → poni
ss   → ss    caress   → caress
s    → ø     cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate relational→ relate
izer→ ize    digitizer → digitize
ator→ ate    operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

J&M/SLP3

Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - *Uygarlastiramadiklarimizdanmissinizcasina*: “(behaving) as if you are among those whom we could not civilize”
  - Uygar `civilized’ + las `become’

    + tir `cause’ + ama `not able’

    + dik `past’ + lar ‘plural’

    + imiz ‘p1pl’ + dan ‘abl’

    + mis ‘past’ + siniz ‘2pl’ + casina ‘as if’

J&M/SLP3

Words

- types and tokens
- morphology
- distributional word vectors
- word sense and lexical semantics

Why is NLP hard?

- ambiguity and variability of linguistic expression:
  - **ambiguity**: one form can mean many things
  - **variability**: many forms can mean the same thing

- one form, multiple meanings → split form

ambiguity

- multiple forms, one meaning → merge forms

variability

Ambiguity

- one form, multiple meanings → split form
  - tokenization (adding spaces):
    - *didn't → did n't*
    - *"Yes?" → " Yes ? "*
  - **today/next week**: word sense disambiguation:
    - *power plant* → *power* $plant_1$
    - *flowering plant* → *flowering* $plant_2$

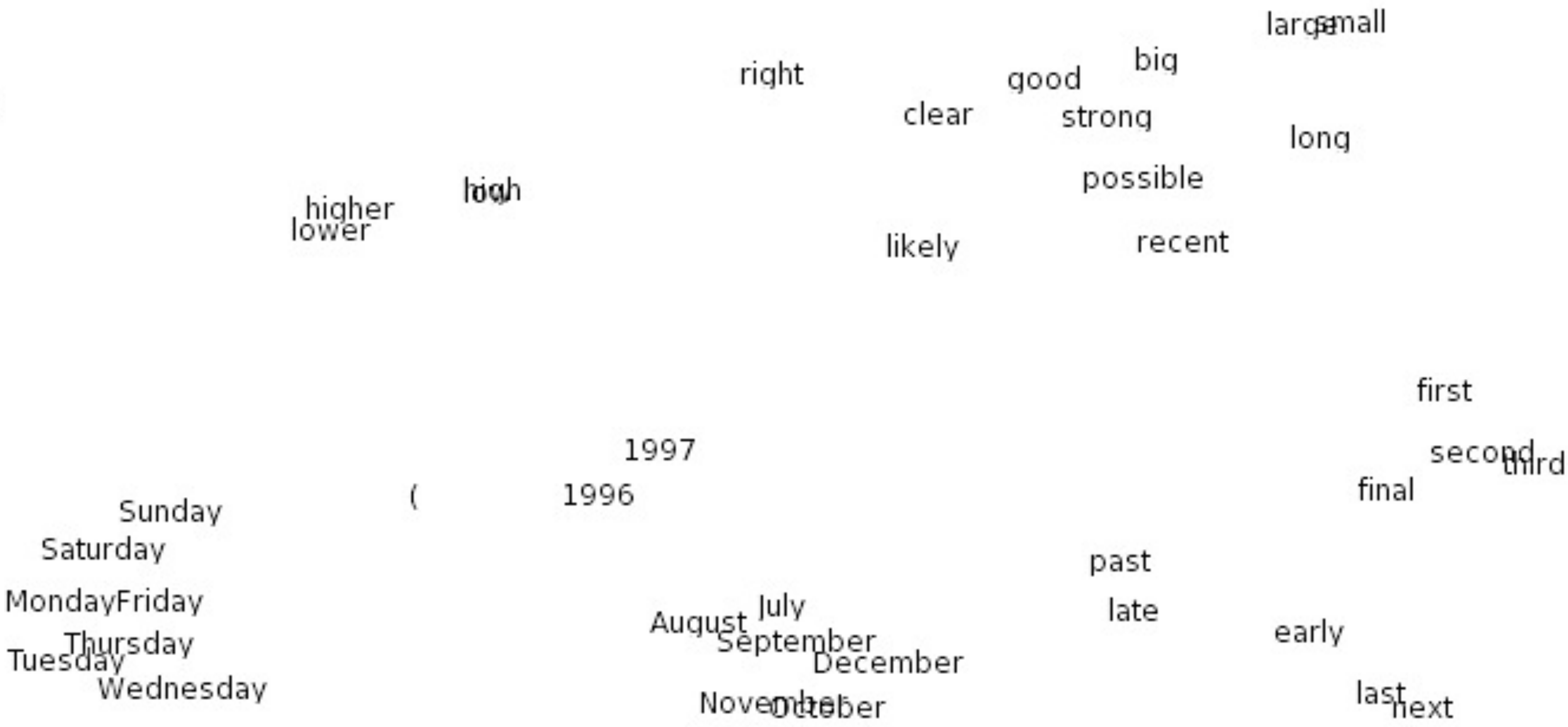
Variability

- multiple forms, one meaning → merge forms
  - tokenization (removing spaces):
    - *New York → NewYork*
  - lemmatization:
    - *walked → walk*
    - *walking → walk*
  - stemming:
    - *automation → automat*
    - *automates → automat*

Variability

- multiple forms, one meaning → merge forms
  - tokenization (removing spaces):
    - *New York* → *NewYork*
  - lemmatization:
    - *walked* → *walk*
    - *walking* → *walk*
  - stemming:
    - *automation* → *automat*
    - *automates* → *automat*
  - **today**: word representations

Vector Representations of Words



t-SNE visualization from Turian et al. (2010)

Word Clusters

Class-Based *n*-gram Models of Natural Language

Peter F. Brown*
Peter V. deSouza*
Robert L. Mercer*
IBM T. J. Watson Research Center

Vincent J. Della Pietra*
Jenifer C. Lai*

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
June March July April January December October November September August
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal

Computational Linguistics, 1992

Why vector models of word meaning? computing the similarity between words

tall is similar to ***height***

question answering:

Q: How ***tall*** *is Mt. Everest?*

A: "The official ***height*** *of Mount Everest is 29029 feet"*

distributional models of meaning
= vector space models of meaning
= vector semantics

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."

J.R. Firth (1957):

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

Warren Weaver (1955):

"But if one lengthens the slit in the opaque mask, until one can see not only the central word in question but also say N words on either side, then if N is large enough one can unambiguously decide the meaning of the central word…"

Intuitions of Distributional Models

- suppose I gave you the following corpus:

  A bottle of ***tesgüino*** is on the table

  Everybody likes ***tesgüino***

  Tesgüino makes you drunk

  We make ***tesgüino*** out of corn.

- what is ***tesgüino***?
- from context, we can guess ***tesgüino*** is an alcoholic beverage like beer
- intuition: two words are similar if they have similar word contexts

Many ways to get word vectors

some based on counting, some based on prediction/learning

some sparse, some dense

some have interpretable dimensions, some don't

shared ideas:

- model meaning of a word by "embedding" it in a vector space
- these word vectors are also called "**embeddings**"

contrast: in traditional NLP, word meaning is represented by a vocabulary index ("word #545")

Distributional Word Vectors

- we'll start with the simplest way to create word vectors:
- count occurrences of context words
  - so, vector for *pineapple* has counts of words in the context of *pineapple* in a dataset
  - one entry in vector for each unique context word
  - stack these vectors for all words in a vocabulary *V* to produce a count matrix *C*
  - *C* is called the **word-context matrix** (or **word-word co-occurrence matrix**)

Counting Context Words

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

Word-Context Matrix

- we showed 4x6, but actual matrix is $|V|$x$|V|$
  - very large, but very **sparse** (mostly zeroes)
  - lots of efficient algorithms for sparse vectors and matrices
  - in your homework assignment, you will sometimes use a different vocabulary V_C for the context, so your matrix will be $|V|$x$|V_C|$

Context Window Size

- size of context window affects word vectors
- in assignment 1, you will explore this both quantitatively and qualitatively

Measuring similarity

- given 2 word vectors, how should we measure their similarity?
- most measure of vector similarity are based on **dot product** (or **inner product**):

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \mathbf{u}^\top \mathbf{v} = \sum_i u_i v_i$$

  – high when vectors have large values in same dimensions

Notation

$$\mathbf{u} = \text{a vector}$$

$$u_i = \text{entry i in the vector}$$

$$\mathbf{u}^\top \mathbf{v} = \text{dot (inner) product}$$

Problem with dot product?

$$\mathbf{u}^\top \mathbf{v} = \sum_i u_i v_i$$

Problem with dot product?

$$\mathbf{u}^\top \mathbf{v} = \sum_i u_i v_i$$

- dot product is larger if vector is longer
- vector length:

$$||\mathbf{u}|| = \sqrt{\sum_i u_i^2}$$

Problem with dot product?

$$\mathbf{u}^\top \mathbf{v} = \sum_i u_i v_i$$

- dot product is larger if vector is longer
- vector length:

$$||\mathbf{u}|| = \sqrt{\sum_i u_i^2}$$

- frequent words → larger counts → larger dot products
- this is bad: we don't want a similarity metric to be overly sensitive to word frequency

J&M/SLP3

Solution: cosine similarity

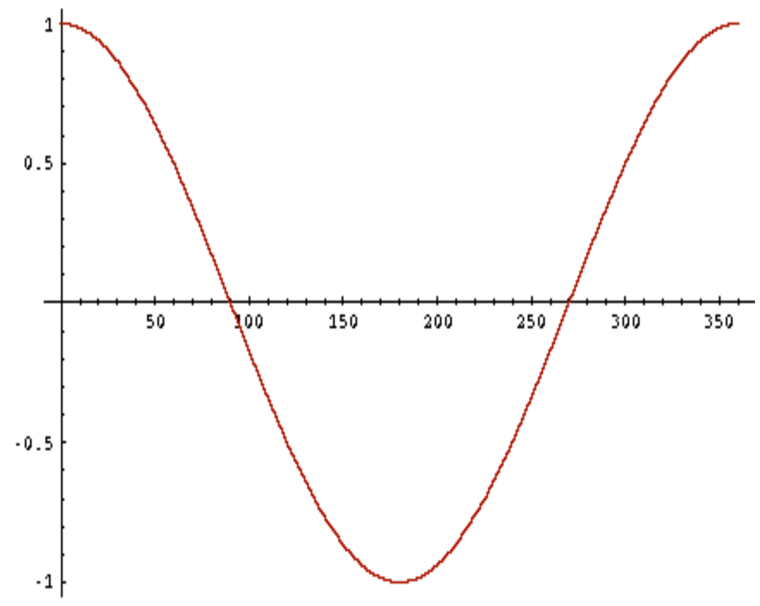
- divide dot product by lengths of the vectors

$$\frac{\mathbf{u}^\top \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||}$$

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

Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal



- word counts are non-negative, so cosine ranges from 0 to 1

- In assignment 1, you should exploit sparsity when counting context words and computing cosine similarities

Problems with raw counts

- raw word counts are not a great measure of association between words
  - why not?
  - very skewed: *the* and *of* are frequent, but not the most discriminative

Top co-occurrence counts with ``cooked''

123	,	13	as
92	and	12	for
79	the	12	food
71	.	11	which
68	<s>	11	that
66	</s>	11	meat
53	in	11	can
39	a	11	by
38	is	10	when
35	of	9	rice
30	with	9	raw
28	are	9	beef
25	to	7	they
23	or	7	their
23	it	7	on
20	(	7	not
19	be	7	from
15	)	6	leaves
14	"	6	has

Top co-occurrence counts with ``cooked''

123	,	13	as
92	and	12	for
79	the	12	**food**
71	.	11	which
68	<s>	11	that
66	</s>	11	**meat**
53	in	11	can
39	a	11	by
38	is	10	when
35	of	9	**rice**
30	with	9	**raw**
28	are	9	**beef**
25	to	7	they
23	or	7	their
23	it	7	on
20	(	7	not
19	be	7	from
15	)	6	**leaves**
14	"	6	has

Problems with raw counts

- raw word counts are not a great measure of association between words
  - why not?
  - very skewed: *the* and *of* are frequent, but not the most discriminative
- rather have a measure that asks whether a context word is **informative** about the center word
  - **pointwise mutual information (PMI)**

Pointwise Mutual Information (PMI)

- do two events *x* and *y* co-occur more often than if they were independent?

$$\text{pmi}(x; y) = \log_2 \frac{p(x, y)}{p(x)p(y)}$$

- here, *x* is the center word and *y* is the word in the context window

- each of these probabilities can be estimated from the counts collected from the corpus

- replace raw counts with pmi scores

Context words of "cooked" with highest PMIs

9.30533	beef	7.66406	chili
8.88418	shrimp	7.56264	rice
8.63397	potatoes	7.56167	soup
8.61946	ate	7.45315	flour
8.56584	dishes	7.43874	steamed
8.50945	eaten	7.43715	crushed
8.4931	beans	7.41193	meals
8.33137	texture	7.39793	digest
8.29489	vegetables	7.39175	rockies
8.25088	soda	7.34773	ramsay
8.20831	meat	7.33211	honey
8.15708	sauce	7.32253	toxicity
8.08345	consuming	7.29057	cared
7.9532	cuisine	7.28626	tomatoes
7.94043	raw	7.27912	boiling
7.78435	curry	7.27769	dal
7.7563	juice	7.27485	citrus
7.74444	vegetable	7.25649	doncaster

Positive Pointwise Mutual Information (PPMI)

- PMI ranges from –infinity to +infinity
- but negative values are problematic:
  - things are co-occurring **less than** we expect by chance
  - unreliable without enormous corpora
- so we sometimes replace negative PMI values by 0, calling it positive PMI (PPMI)

Alternative to PPMI

- **tf-idf**: (that's a hyphen not a minus sign)
- product of two factors:
  - **term frequency** (TF; Luhn, 1957): count of word (or possibly log of count)
  - **inverse document frequency** (IDF; Sparck Jones, 1972)
    - *N:* total number of documents
    - df(*x*): # of documents with word *x*

$$\mathrm{idf}(x) = \log \frac{N}{\mathrm{df}(x)}$$

J&M/SLP3

How should we evaluate word vectors?

WordSim353

(Finkelstein et al., 2002)

word pair		similarity
journey	voyage	
king	queen	
computer	software	
law	lawyer	
forest	graveyard	
rooster	voyage	

WordSim353

(Finkelstein et al., 2002)

Instructions:

Assign a numerical similarity score between 0 and 10
(0 = words are totally unrelated,
10 = words are VERY closely related).

computer	software	
law	lawyer	
forest	graveyard	
rooster	voyage	

WordSim353

(Finkelstein et al., 2002)

Instructions:

Assign a numerical similarity score between 0 and 10
(0 = words are totally unrelated,
10 = words are VERY closely related).

When estimating similarity of antonyms, consider them "similar" (i.e., belonging to the same domain or representing features of the same concept), **rather than "dissimilar"**.

forest	graveyard	
rooster	voyage	

WordSim353

(Finkelstein et al., 2002)

word pair		similarity
journey	voyage	9.3
king	queen	8.6
computer	software	8.5
law	lawyer	8.4
forest	graveyard	1.9
rooster	voyage	0.6

SimLex-999

(Hill et al., 2014)

word pair		similarity
insane	crazy	9.6
attorney	lawyer	9.4
author	creator	8.0
diet	apple	1.2
new	ancient	0.2

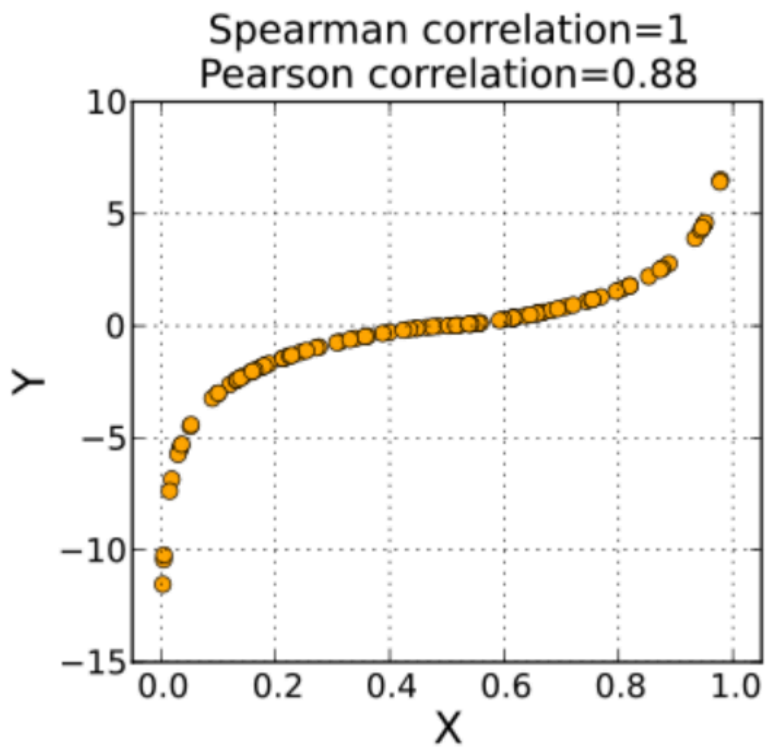
measures **paraphrastic** similarity:

two words are “similar” if they have similar meanings

- there are many word similarity datasets
- some focus on topical relatedness, others focus on similarity in meaning
- in assignment 1, you will evaluate your word vectors using MEN (relatedness) and SimLex-999 (meaning)

Evaluation Metrics for Word Similarity

- Spearman rank correlation coefficient
- measures correlation between two variables:
  - variable 1: human-annotated similarities for word pairs
  - variable 2: cosine similarities computed with your word vectors for the same word pairs



Wikipedia