

Text Analysis

Everything Data
CompSci 216 Spring 2015



DUKE
COMPUTER SCIENCE

Announcements (Wed. Feb. 11)

- HW 5 will be posted by tomorrow (Thu) morning.
- Project: More information in the lab on Monday!

Outline

- Basic Text Processing
 - Word Counts
 - Tokenization
 - Pointwise Mutual Information
 - Normalization
 - Stemming

Outline

- Basic Text Processing
- Finding Salient Tokens
 - TFIDF scoring

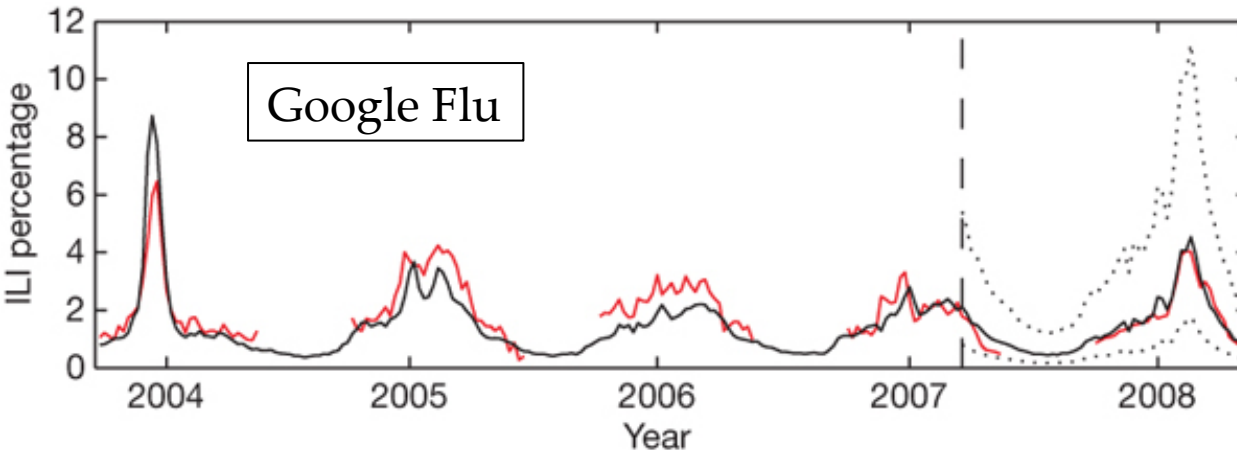
Outline

- Basic Text Processing
- Finding Salient Tokens
- Document Similarity & Keyword Search
 - Vector Space Model & Cosine Similarity
 - Inverted Indexes

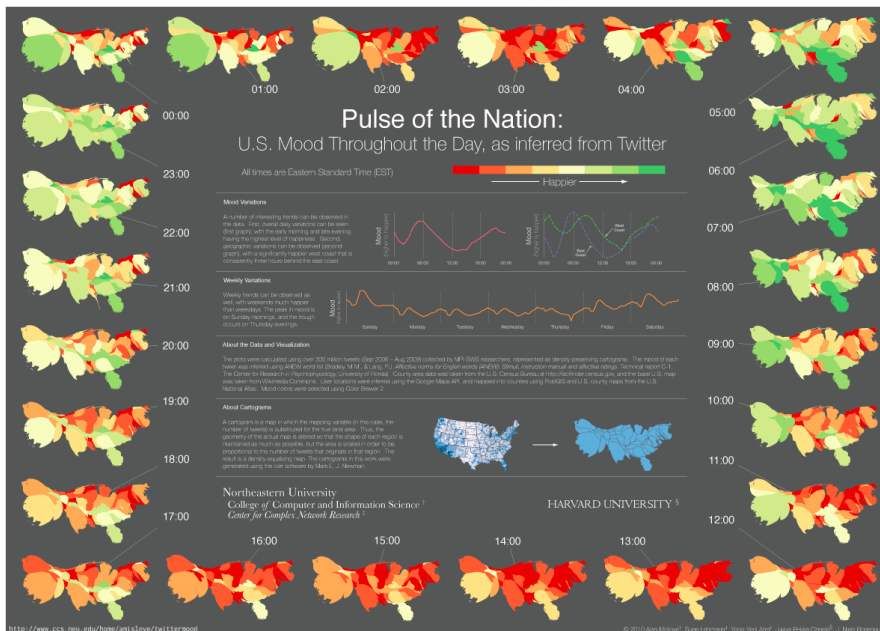
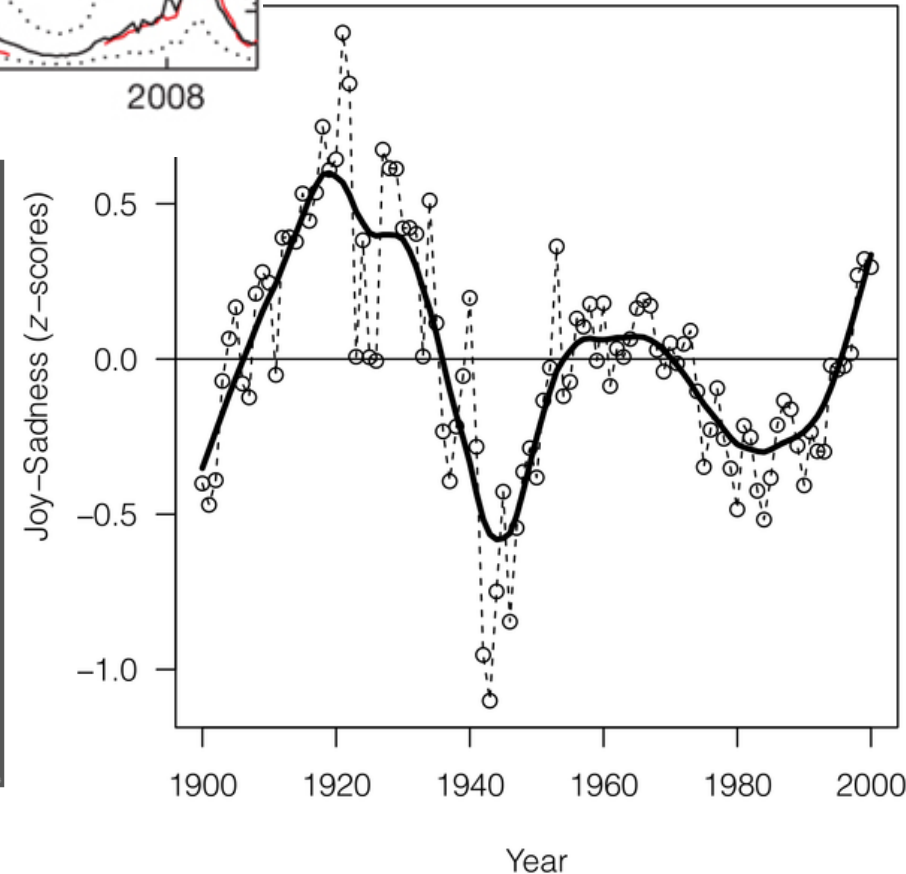
Outline

- Basic Text Processing
 - Word Counts
 - Tokenization
 - Pointwise Mutual Information
 - Normalization
 - Stemming
- Finding Salient Tokens
- Document Similarity & Keyword Search

Basic Query: Word Counts



Emotional Trends



<http://www.ccs.neu.edu/home/amislove/twittermood/>

Basic Query: Word Counts

- How many times does each word appear in the document?

Problem 1

- What is a word?
 - I'm ... 1 word or 2 words {I'm} or {I, am}
 - State-of-the-art ... 1 word or 4 words
 - San Francisco ... 1 or 2 words
- Other Languages
 - French: l'ensemble
 - German: freundschaftsbezeugungen
 - Chinese: 這是一個簡單的句子 (no spaces)

This, one, easy, sentence

Solution: Tokenization

- For English:
 - Splitting the text on non alphanumeric characters is a reasonable way to find individual words.
- But will split words like “San Francisco” and “state-of-the-art”
- For other languages:
 - Need more sophisticated algorithms for identifying words.

Finding simple phrases

- Want to find pairs of tokens that always occur together (**co-occurrence**)
- Intuition:
Two tokens are co-occurrent if they appear together more often than “*random*”
 - *More often than monkeys typing English words*



<http://tumblr.co/ZiEAFywGEckV>

Formalizing “*more often than random*”

- **Language Model**

- Assigns a probability $P(x_1, x_2, \dots, x_k)$ to any sequence of tokens.
- More common sequences have a higher probability
- Sequences of length 1: **unigrams**
- Sequences of length 2: **bigrams**
- Sequences of length 3: **trigrams**
- Sequences of length n : **n-grams**

Formalizing “*more often than random*”

- Suppose we have a language model
 - $P(\text{“San Francisco”})$ is the probability that “San” and “Francisco” occur together (and in that order) in the language.

Formalizing “*random*”: Bag of words

- Suppose we only have access to the unigram language model
 - *Think: all unigrams in the language thrown into a bag with counts proportional to their $P(x)$*
 - *Monkeys drawing words at random from the bag*
 - $P(\text{“San”}) \times P(\text{“Francisco”})$ is the probability that “San Francisco” occurs together in the (random) unigram model

Formalizing “*more often than random*”

- **Pointwise Mutual Information:**

$$\text{PMI}(x_1, x_2) = \log_2 \frac{P(x_1, x_2)}{P(x_1) \cdot P(x_2)}$$

- Positive PMI suggests word co-occurrence
- Negative PMI suggests words don’t appear together

What is $P(x)$?

- *“Suppose we have a language model ...”*
- Idea: Use counts from a large corpus of text to compute the probabilities

What is $P(x)$?

- Unigram: $P(x) = \text{count}(x) / N$
 - $\text{count}(x)$ is the number of times token x appears
 - N is the total # tokens.
- Bigram: $P(x_1, x_2) = \text{count}(x_1, x_2) / N$
 - $\text{count}(x_1, x_2) = \#$ times sequence (x_1, x_2) appears
- Trigram: $P(x_1, x_2, x_3) = \text{count}(x_1, x_2, x_3) / N$
 - $\text{count}(x_1, x_2, x_3) = \#$ times sequence (x_1, x_2, x_3) appears

What is $P(x)$?

- “*Suppose we have a language model ...*” ✓
- Idea: Use counts from a **large corpus** of text to compute the probabilities

Large text corpora

- Corpus of Contemporary American English
 - <http://corpus.byu.edu/coca/>
- Google N-gram viewer
 - <https://books.google.com/ngrams>

Summary of Problem 1

- Word tokenization can be hard
- Space/non-alphanumeric words may oversplit the text
- We can find co-occurrent tokens:
 - Build a **language model** from a large corpus
 - Check whether the pair of tokens appear more often than random using **pointwise mutual information**.

Language models ...

- ... have many many applications
 - Tokenizing long strings
 - Word/query completion/suggestion
 - Spell checking
 - Machine translation
 - ...

Problem 2

- A word may be represented in many forms
 - car, cars, car's, cars' → car
 - automation, automatic → automate
- Lemmatization: Problem of finding the correct dictionary headword form

Solution: Stemming

- Words are made up of
 - Stems: core word
 - Affixes: modifiers added (often with grammatical function)
- Stemming: reduce terms to their stems by crudely chopping off affixes
 - automation, automatic → automat

Porter's algorithm for English

- Sequences of rules applied to words

Example rule sets:

```
/(.*)sses$/ → \1ss  
/(.*)ies$/ → \1i  
/(.*)ss$/ → \1s  
/(.*[^s])s$/ → \1
```

```
/(.*[aeiou]+.*)ing$/ → \1  
/(.*[aeiou]+.*)ed$/ → \1
```


Any other problems?

- Same words that mean different things
 - Florence the person vs Florence the city
 - Paris Hilton (person or hotel)
- Abbreviations
 - I.B.M vs International Business Machines
- Different words meaning same thing
 - Big Apple vs New York
- ...

Any other problems?

- Same words that mean different things
 - *Word Sense Disambiguation*
- Abbreviations
 - *Translations*
- Different words meaning same thing
 - *Named Entity Recognition & Entity Resolution*
- ...

Outline

- Basic Text Processing
- Finding Salient Tokens
 - TFIDF scoring
- Document Similarity & Keyword Search

Summarizing text




Belinelli's late jumper gives **Popovich his 1000th career w...**
Yahoo Sports (blog) - 4 hours ago
 Gregg **Popovich** of the San Antonio Spurs has already established himself ... Nevertheless, it's pretty cool and rare any time a coach hits **1,000** ...


Recommended Reviews for Udupi Cafe

Reviews Matching:

[Search Reviews](#)

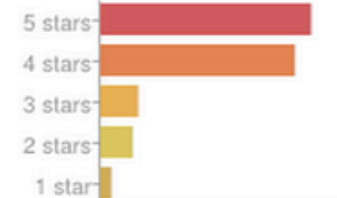
Review Highlights [What's this?](#)

 "They are absolutely delicious try the **dosa** and mango lasy yum."
 In 27 reviews

 "Excellent **buffet** with an awesome selection."
 In 40 reviews

 "The fresh **coconut chutney** rocks my world, as does the service at..."
 In 6 reviews

Rating Distribution | [Trend](#)



Finding salient tokens (words)

- Most frequent tokens?



Rank	Word	Rank	Word	Rank	Word	Rank	Word	Rank	Word
1	the	21	this	41	so	61	people	81	back
2	be	22	but	42	up	62	into	82	after
3	to	23	his	43	out	63	year	83	use
4	of	24	by	44	if	64	your	84	two
5	and	25	from	45	about	65	good	85	how
6	a	26	they	46	who	66	some	86	our
7	in	27	we	47	get	67	could	87	work
8	that	28	say	48	which	68	them	88	first
9	have	29	her	49	go	69	see	89	well
10	I	30	she	50	me	70	other	90	way
11	it	31	or	51	when	71	than	91	even
12	for	32	an	52	make	72	then	92	new
13	not	33	will	53	can	73	now	93	want
14	on	34	my	54	like	74	look	94	because
15	with	35	one	55	time	75	only	95	any
16	he	36	all	56	no	76	come	96	these
17	as	37	would	57	just	77	its	97	give
18	you	38	there	58	him	78	over	98	day
19	do	39	their	59	know	79	think	99	most
20	at	40	what	60	take	80	also	100	us

Document Frequency

- Intuition: Uninformative words appear in many documents (not just the one we are concerned about)
- Salient word:
 - High count within the document
 - Low count across documents

TF•IDF score

- Term Frequency (TF):

$$TF(x) = \log_{10}(1 + c(x)) \text{ or } c(x)$$

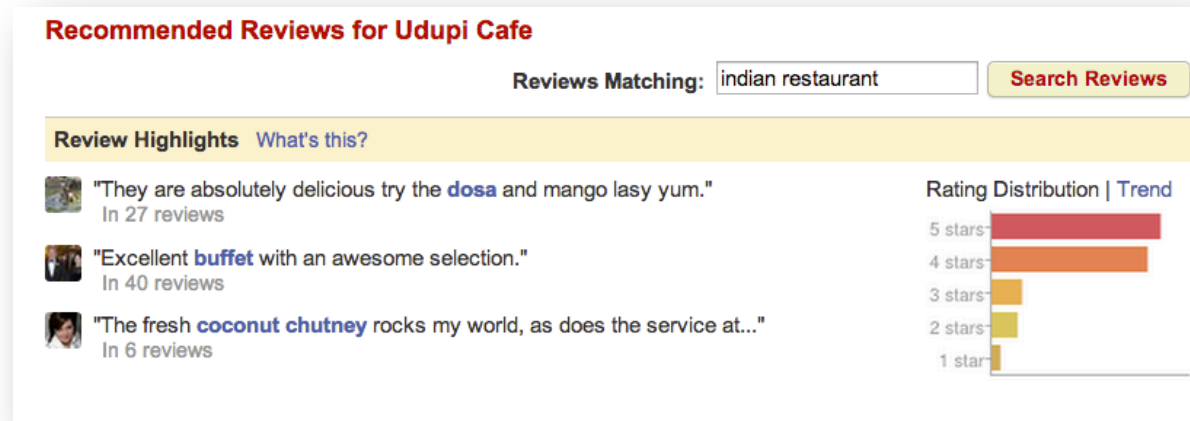
$c(x)$ is # times x appears in the document

- Inverse Document Frequency (IDF):

$$IDF(x) = \log_{10} \left(\frac{N_{docs}}{DF(x)} \right)$$

$DF(x)$ is the number of documents x appears in.

Back to summarization



- Simple heuristic:
 - Pick sentences $S = \{x_1, x_2, \dots, x_k\}$ with the highest:

$$\text{Salience}(S) = \frac{1}{|S|} \sum_{x \in S} \text{TF}(x) \cdot \text{IDF}(x)$$

Outline

- Basic Text Processing
- Finding Salient Tokens
- Document Similarity & Keyword Search
 - Vector Space Model & Cosine Similarity
 - Inverted Indexes

Document Similarity



Belinelli's late jumper gives Popovich his 1000th career w...

Yahoo Sports (blog) - 4 hours ago

Gregg **Popovich** of the San Antonio Spurs has already established himself ... Nevertheless, it's pretty cool and rare any time a coach hits **1,000** ...

Spurs' Gregg Popovich becomes 9th NBA coach to win 1000 games

SI.com - 20 hours ago

SVG: Popovich's 1000 wins 'a great accomplishment'

Detroit Free Press - 2 minutes ago

Gregg Popovich Wins 1000th Game with Milestones Ahead & Other ...

In-Depth - Bleacher Report - 18 hours ago

Six things to know about Gregg Popovich's 1000th win

Blog - Washington Post (blog) - 20 hours ago

Raptors Beat Spurs, Deny Popovich 1000th Win

In-Depth - ABC News - Feb 8, 2015

Vector Space Model

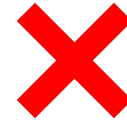
- Let $V = \{x_1, x_2, \dots, x_n\}$ be the set of all tokens (across all documents)
- A document is a n -dimensional vector

$$D = [w_1, w_2, \dots, w_n]$$

where $w_i = \text{TFIDF}(x_i, D)$

Distance between documents

- Euclidean distance

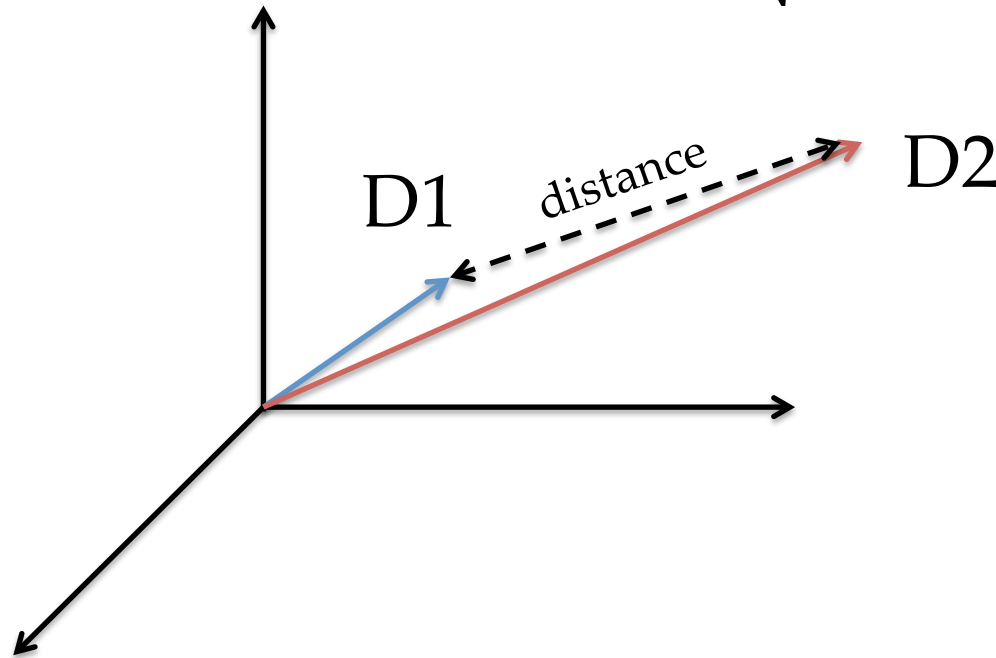


$$D1 = [w_1, w_2, \dots, w_n]$$

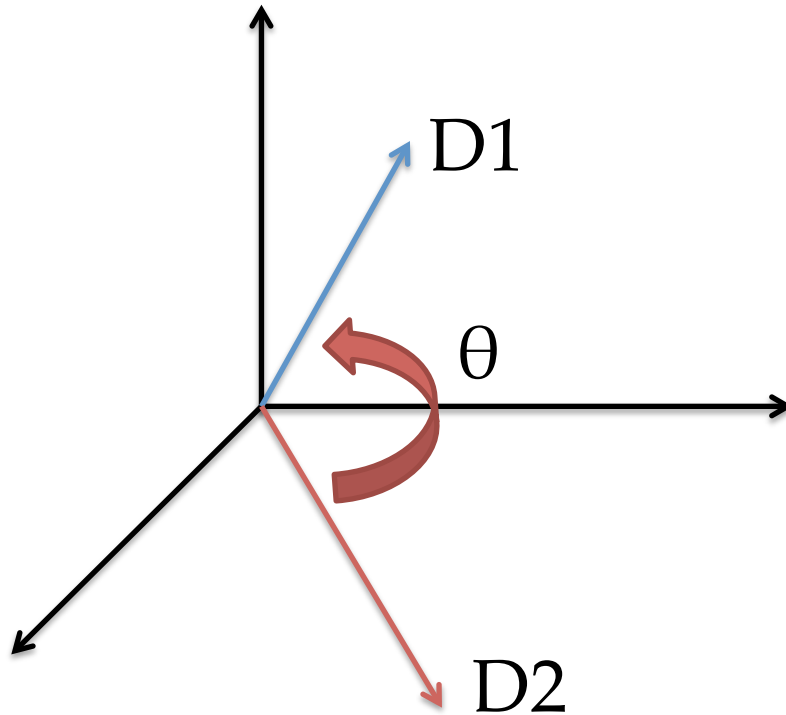
$$D2 = [y_1, y_2, \dots, y_n]$$

$$d(D1, D2) = \sqrt{\sum_i (w_i - y_i)^2}$$

- Why?



Cosine Similarity



$$d(D1, D2) = \cos(\theta)$$

Cosine Similarity

- $D1 = [w_1, w_2, \dots, w_n]$
- $D2 = [y_1, y_2, \dots, y_n]$

$$d(D1, D2) = \frac{\sum_i w_i \cdot y_i}{\sqrt{\sum_i w_i^2} \sqrt{\sum_i y_i^2}}$$

Dot Product

L2 Norm

Keyword Search

A horizontal search bar with a light gray background. It features a white input field with a blue border and a blue button on the right containing a white magnifying glass icon.

- How to find documents that are similar to a keyword query?
- Intuition: Think of the query as another (very short) document

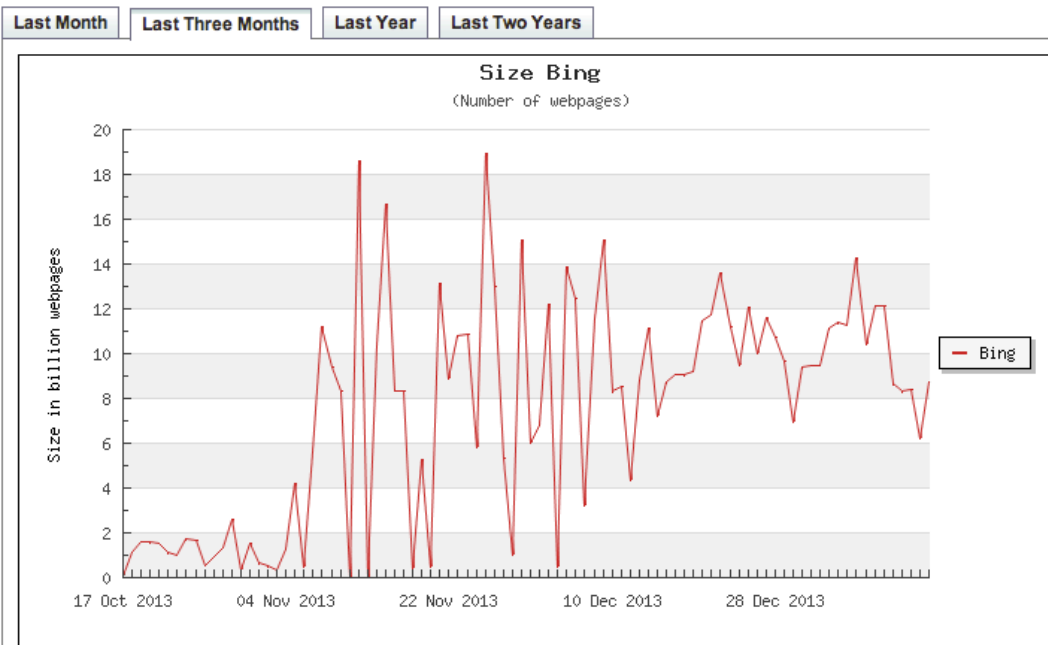
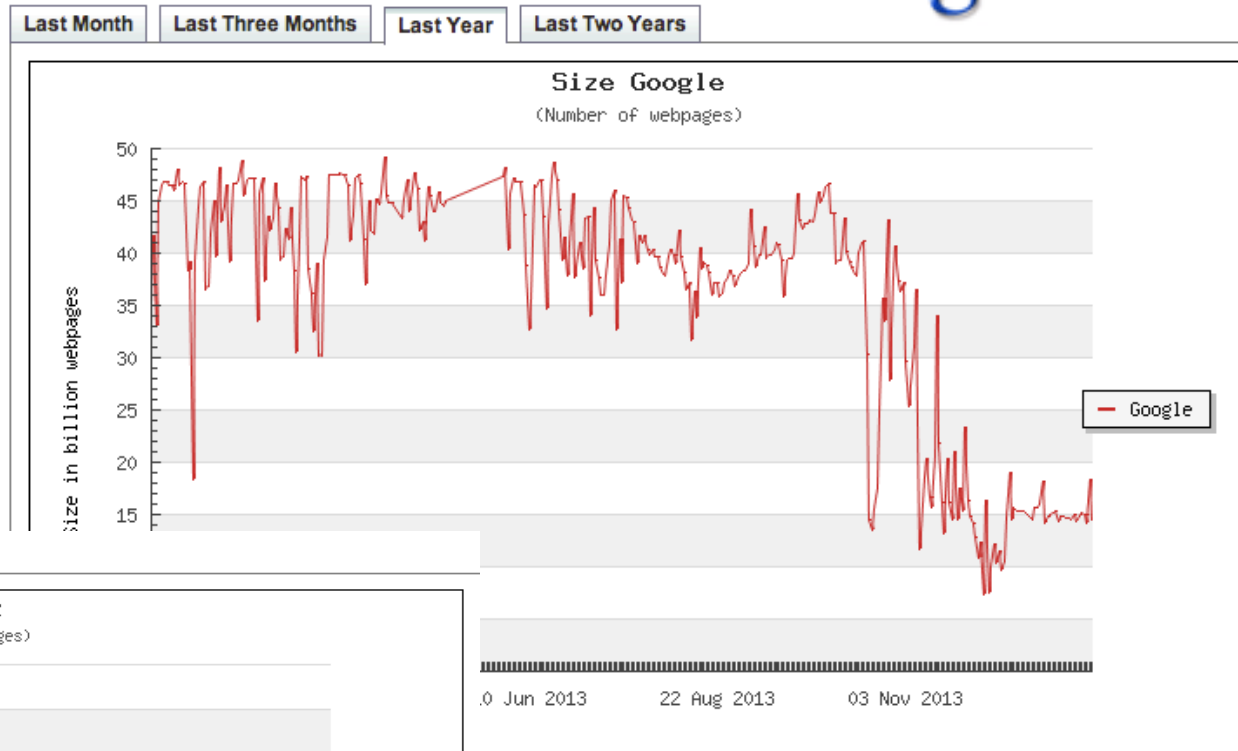
Keyword Search

- Simple Algorithm

For every document D in the corpus
 Compute $d(q, D)$

Return the top- k highest scoring documents

Does it scale?



<http://www.worldwidewebsize.com/>

Inverted Indexes

- Let $V = \{x_1, x_2, \dots, x_n\}$ be the set of **all** token
- For each token, store
 <token, sorted list of documents token
 appears in>
 - <“caeser”, [1,3,4,6,7,10,...]>
- How does this help?

Using Inverted Lists

- Documents containing “caesar”
 - Use the inverted list to find documents containing “caesar”
- What additional information should we keep to compute similarity between the query and documents?

Using Inverted Lists

- Documents containing “caesar” AND “cleopatra”
 - Return documents in the intersection of the two inverted lists.
 - Why is inverted list sorted on document id?
- OR? NOT?
 - Union and difference, resp.

Many other things in a modern search engine ...

- Maintain positional information to answer phrase queries
- Scoring is not only based on token similarity
 - Importance of Webpages: PageRank (in later classes)
- User Feedback
 - Clicks and query reformulations

Summary

- Word counts are very useful when analyzing text
 - Need good algorithms for **tokenization**, **stemming** and other **normalizations**
- Algorithm for finding word co-occurrence
 - **Language Models**
 - **Pointwise Mutual Information**

Summary (contd.)

- Raw counts are not sufficient to find salient tokens in a document
 - Term Frequency x Inverse Document Frequency (**TFIDF**) scoring
- Keyword Search
 - Use **Cosine Similarity** over TFIDF scores to compute similarity
 - Use **Inverted Indexes** to speed up processing.