Phrase Finding

William W. Cohen

Outline of the course

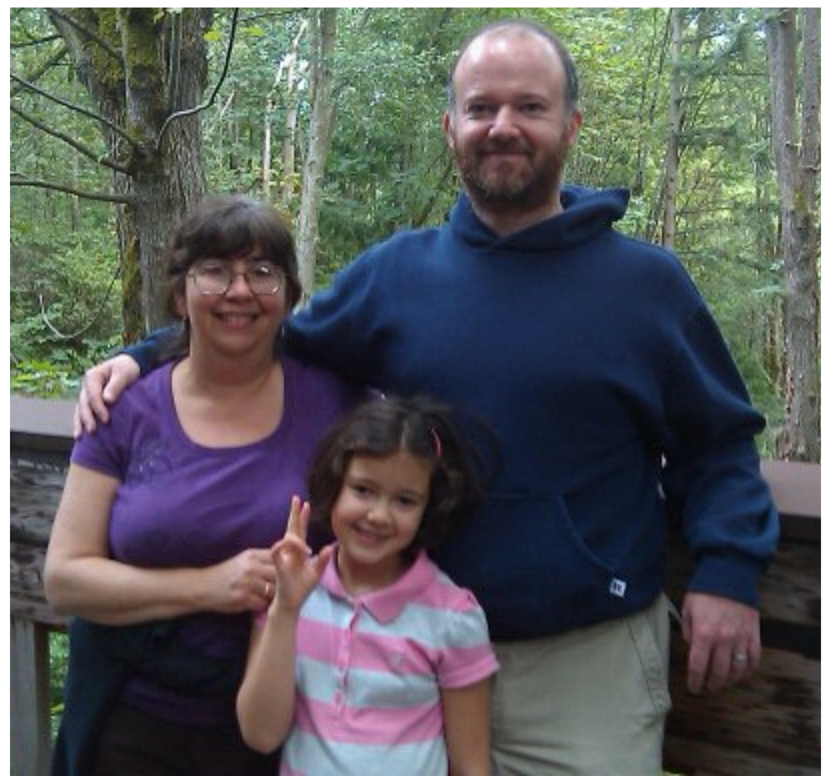
- Week 1: review and a fruit fly (algorithm to to study)
  - Time complexity, cost of operations, and Naïve Bayes v1
- Week 2-4: scaling and parallelizing Naïve Bayes
  - Computational paradigms: "stream and sort", "map-reduce", high-level "data-flow" operations
  - Tasks you will do:
    - Training Naïve Bayes on a large vocabulary (HW1)
    - Parallel Naïve Bayes with Hadoop (HW2; low-level)
    - Parallel testing of a large-vocabulary linear classifier (HW3; high-level dataflow language)
  - We'll go back and forth between paradigms
    - You can't forget about the low-level stuff (yet!)
  - Other tasks we will talk about:
    - Computing IDF, Rocchio's algorithm, **phrase finding**
    - "Soft joins"

A Language Model Approach to Keyphrase Extraction

Takashi Tomokiyo and Matthew Hurst
Applied Research Center
Intelliseek, Inc.
Pittsburgh, PA 15213
{ttomokiyo,mhurst}@intelliseek.com

A Language Model Approach to Keyphrase Extraction

Takashi Tomokiyo and Matthew Hurst
Applied Research Center
Intelliseek, Inc.
Pittsburgh, PA 15213
{ttomokiyo,mhurst}@intelliseek.com



ACL Workshop 2003

1	civic hybrid	21	mustang gt
2	honda civic hybrid	22	ford escape
3	toyota prius	23	steering wheel
4	electric motor	24	toyota prius today
5	honda civic	25	electric motors
6	fuel cell	26	gasoline engine
7	hybrid cars	27	internal combustion engine
8	honda insight	28	gas engine
9	battery pack	29	front wheels
10	sports car	30	key sense wire
11	civic si	31	civic type r
12	hybrid car	32	test drive
13	civic lx	33	street race
14	focus fcv	34	united states
15	fuel cells	35	hybrid powertrain
16	hybrid vehicles	36	rear bumper
17	tour de sol	37	ford focus
18	years ago	38	detroit auto show
19	daily driver	39	parking lot
20	jetta tdi	40	rear wheels

Figure 1: Top 40 keyphrases automatically extracted from messages relevant to "*civic hybrid*" using our system

Why phrase-finding?

- There are lots of phrases
- There's not supervised data
- It's hard to articulate
  - What makes a phrase a phrase, *vs* just an n-gram?
    - a phrase is independently meaningful ("test drive", "red meat") or not ("are interesting", "are lots")
  - What makes a phrase interesting?

The breakdown: what makes a good phrase

- Two properties:
  - Phraseness: “the degree to which a given word sequence is considered to be a phrase”
    - Statistics: how often words co-occur together vs separately
  - Informativeness: “how well a phrase captures or illustrates the key ideas in a set of documents” – something novel and important **relative to a domain**
    - Background corpus and foreground corpus; how often phrases occur in each

"Phraseness"$_1$ – based on BLRT

- Binomial Ratio Likelihood Test (BLRT):
  - Draw samples:
    - n_1 draws, k_1 successes
    - n_2 draws, k_2 successes
    - Are they from one binominal (i.e., k_1/n_1 and k_2/n_2 were different due to chance) or from two distinct binomials?
  - Define
    - $p_1=k_1/n_1$, $p_2=k_2/n_2$, $p=(k_1+k_2)/(n_1+n_2)$,
    - $L(p,k,n) = p^k(1-p)^{n-k}$

$$BLRT(n_1,k_1,n_2,k_2) = \frac{L(p_1,k_1,n_1)L(p_2,k_2,n_2)}{L(p,k_1,n_1)L(p,k_2,n_2)}$$

“Phraseness”$_1$ – based on BLRT

- Binomial Ratio Likelihood Test (BLRT):
  - Draw samples:
    - n_1 draws, k_1 successes
    - n_2 draws, k_2 successes
    - Are they from one binominal (i.e., k_1/n_1 and k_2/n_2 were different due to chance) or from two distinct binomials?
  - Define
    - $p_i=k_i/n_i$, $p=(k_1+k_2)/(n_1+n_2)$,
    - $L(p,k,n) = p^k(1-p)^{n-k}$

$$BLRT(n_1,k_1,n_2,k_2) = 2\log\frac{L(p_1,k_1,n_1)L(p_2,k_2,n_2)}{L(p,k_1,n_1)L(p,k_2,n_2)}$$

"Phraseness"$_1$ – based on BLRT

– Define
  - $p_i=k_i/n_i$, $p=(k_1+k_2)/(n_1+n_2)$,
  - $L(p,k,n) = p^k(1-p)^{n-k}$

Phrase x y: $W_1=x$ ^ $W_2=y$

$$\varphi_p(n_1,k_1,n_2,k_2) = 2\log\frac{L(p_1,k_1,n_1)L(p_2,k_2,n_2)}{L(p,k_1,n_1)L(p,k_2,n_2)}$$

		comment
k_1	$C(W_1=x \wedge W_2=y)$	how often bigram *x y* occurs in corpus C
n_1	$C(W_1=x)$	how often word *x* occurs in corpus C
k_2	$C(W_1\neq x \wedge W_2=y)$	how often *y* occurs in C after a non-*x*
n_2	$C(W_1\neq x)$	how often a non-*x* occurs in C

Does *y* occur at the same frequency after *x* as in other positions?

"Informativeness"$_1$ – based on BLRT

– Define

- $p_i=k_i/n_i$, $p=(k_1+k_2)/(n_1+n_2)$,
- $L(p,k,n) = p^k(1-p)^{n-k}$

Phrase $x\ y$: $W_1=x \wedge W_2=y$ and two corpora, C and B

$$\varphi_i(n_1,k_1,n_2,k_2) = 2\log\frac{L(p_1,k_1,n_1)L(p_2,k_2,n_2)}{L(p,k_1,n_1)L(p,k_2,n_2)}$$

		comment
k_1	$C(W_1=x \wedge W_2=y)$	how often bigram *x y* occurs in corpus C
n_1	$C(W_1=* \wedge W_2=*)$	how many bigrams in corpus C
k_2	$B(W_1=x \wedge W_2=y)$	how often *x y* occurs in **background corpus**
n_2	$B(W_1=* \wedge W_2=*)$	how many bigrams in background corpus

Does x *y* occur at the same frequency in both corpora?

The breakdown: what makes a good phrase

- "Phraseness" and "informativeness" are then combined with a tiny classifier, tuned on labeled data.

$$\varphi = \frac{1}{1 + \exp(-a\varphi_p - b\varphi_i + c)}$$

$$\left(\log \frac{p}{1-p} = s\right) \Leftrightarrow \left(p = \frac{1}{1+e^s}\right)$$

  - Background corpus: 20 newsgroups dataset (20k messages, 7.4M words)
  - Foreground corpus: rec.arts.movies.current-films June-Sep 2002 (4M words)
  - Results?

1	message news	16	sixth sense
2	minority report	17	hey kids
3	star wars	18	gaza man
4	john harkness	19	lee harrison
5	derek janssen	20	years ago
6	robert frenchu	21	julia roberts
7	sean o'hara	22	national guard
8	box office	23	bourne identity
9	dawn taylor	24	metrotoday www.zap2it.com
10	anthony gaza	25	starweek magazine
11	star trek	26	eric chomko
12	ancient race	27	wilner starweek
13	scooby doo	28	tim gueguen
14	austin powers	29	jodie foster
15	home.attbi.com hey	30	johnnie kendricks

The breakdown: what makes a good phrase

- Two properties:
  - Phraseness: "the degree to which a given word sequence is considered to be a phrase"
    - Statistics: how often words co-occur together vs separately
  - Informativeness: "how well a phrase captures or illustrates the key ideas in a set of documents" – something novel and important relative to a domain
    - Background corpus and foreground corpus; how often phrases occur in each
  - Another intuition: our goal is to compare distributions and see how **different** they are:
    - Phraseness: estimate $x\ y$ with bigram model or unigram model
    - Informativeness: estimate with foreground vs background corpus

The breakdown: what makes a good phrase

- Another intuition: our goal is to compare distributions and see how **different** they are:
  - Phraseness: estimate *x y* with bigram model or unigram model
  - Informativeness: estimate with foreground vs background corpus
- To compare distributions, use KL-divergence

$$D(p \,||\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

"Pointwise KL divergence"

$$\delta_{\mathbf{w}}(p \,||\, q) \stackrel{\text{def}}{=} p(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})}$$

The breakdown: what makes a good phrase

– To compare distributions, use KL-divergence

$$D(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

"Pointwise KL divergence"

$$\delta_{\mathbf{w}}(p \parallel q) \stackrel{\text{def}}{=} p(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})}$$

Phraseness: difference between bigram and unigram language model in foreground

$$\delta_{\mathbf{w}}(LM^N_{\mathrm{fg}} \parallel LM^1_{\mathrm{fg}})$$

Bigram model: P(x y)=P(x)P(y | x)

Unigram model: P(x y)=P(x)P(y)

The breakdown: what makes a good phrase

– To compare distributions, use KL-divergence

$$D(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

"Pointwise KL divergence"

$$\delta_{\mathbf{w}}(p \parallel q) \stackrel{\text{def}}{=} p(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})}$$

Informativeness: difference between foreground and background models

$$\delta_{\mathbf{w}}(LM_{\text{fg}}^{N} \parallel LM_{\text{bg}}^{N}), \text{ or}$$
$$\delta_{\mathbf{w}}(LM_{\text{fg}}^{1} \parallel LM_{\text{bg}}^{1})$$

Bigram model: P(x y)=P(x)P(y|x)

Unigram model: P(x y)=P(x)P(y)

$$\delta_{\mathbf{w}}(LM_{\text{fg}}^{N} \parallel LM_{\text{bg}}^{1})$$

The breakdown: what makes a good phrase

– To compare distributions, use KL-divergence

$$D(p \,||\, q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

"Pointwise KL divergence"

$$\delta_{\mathbf{w}}(p \,||\, q) \stackrel{\text{def}}{=} p(\mathbf{w}) \log \frac{p(\mathbf{w})}{q(\mathbf{w})}$$

Bigram model: P(x y)=P(x)P(y | x)

Unigram model: P(x y)=P(x)P(y)

Combined: difference between foreground bigram model and background unigram model

$$\delta_{\mathbf{w}}(LM^N_{\text{fg}} \,||\, LM^1_{\text{bg}})$$

$\longleftarrow$ *informativeness* $\longrightarrow$

$\uparrow$ *phraseness* $\downarrow$

$$\begin{tikzcd} LM^N_{\text{fg}} \arrow[r, dash] \arrow[d, dash] & LM^N_{\text{bg}} \arrow[d, dash] \\ LM^1_{\text{fg}} \arrow[r, dash] & LM^1_{\text{bg}} \end{tikzcd}$$

The breakdown: what makes a good phrase

– To compare distributions, use KL-divergence

Subtle advantages:

- BLRT scores "more frequent in foreground" and "more frequent in background" symmetrically, pointwise KL does not.
- Phrasiness and informativeness scores are more comparable – straightforward combination w/o a classifier is reasonable.
- Language modeling is well-studied:
  - extensions to n-grams, smoothing methods, …
  - we can build on this work in a modular way

Combined: difference between foreground bigram model and background unigram model

$$\delta_{\mathbf{w}}(LM^N_{\text{fg}} \parallel LM^1_{\text{bg}})$$

$$\longleftarrow \textit{informativeness} \longrightarrow$$

$$\textit{phraseness} \updownarrow \quad \begin{array}{ccc} LM^N_{\text{fg}} & \text{———} & LM^N_{\text{bg}} \\ | & & | \\ LM^1_{\text{fg}} & \text{———} & LM^1_{\text{bg}} \end{array}$$

Pointwise KL, combined

1	message news	16	hey kids
2	minority report	17	years ago
3	star wars	18	gaza man
4	john harkness	19	sixth sense
5	robert frenchu	20	lee harrison
6	derek janssen	21	julia roberts
7	box office	22	national guard
8	sean o'hara	23	bourne identity
9	dawn taylor	24	metrotoday www.zap2it.com
10	anthony gaza	25	starweek magazine
11	star trek	26	eric chomko
12	ancient race	27	wilner starweek
13	home.attbi.com hey	28	tim gueguen
14	scooby doo	29	jodie foster
15	austin powers	30	kevin filmnutboy

Why phrase-finding?

- Phrases are where the standard supervised "bag of words" representation starts to break.
- There's not supervised data, so it's hard to see what's "right" and why
- It's a nice example of using unsupervised signals to solve a task that could be formulated as supervised learning
- It's a nice level of complexity, if you want to do it in a scalable way.

Implementation

- Request-and-answer pattern
  - Main data structure: tables of key-value pairs
    - *key* is a phrase *x y*
    - *value* is a mapping from a attribute names (like *phraseness*, *freq-in-B, …*) to numeric values.
  - Keys and values are just strings
  - We'll operate mostly by sending messages to this data structure and getting results back, or else streaming thru the whole table
  - For really big data: we'd also need tables where *key* is a word and *val* is set of attributes of the word *(freq-in-B, freq-in-C, …)*

Key	Value
`old man`	`freq(B)=10,freq(C)=13,informativeness=1.3,phrasiness=740`
`bad service`	`freq(B)=8,freq(C)=25,informativeness=560,phrasiness=254`
...	...

Generating and scoring phrases: 1

- Stream through **foreground** corpus and count events "$W_1=x$ ^ $W_2=y$" the same way we do in training naive Bayes: stream-and sort and accumulate deltas (a "sum-reduce")
  - Don't bother generating boring phrases (e.g., crossing a sentence, contain a stopword, …)
- Then stream through the output and convert to *phrase, attributes-of-phrase* records with one attribute: *freq-in-C=n*
- Stream through foreground corpus and count events "$W_1=x$" in a (memory-based) hashtable….
- This is enough* to compute phrasiness:
  - $\psi_p(x\ y) = f(\ \textit{freq-in-C}(x), \textit{freq-in-C}(y), \textit{freq-in-C}(x\ y))$
- …so you can do that with a scan through the phrase table that **adds** an extra attribute (holding word frequencies in memory).

* actually you also need total # words and total #phrases….

Generating and scoring phrases: 2

- Stream through **background** corpus and count events "$W_1=x$ ^ $W_2=y$" and convert to *phrase, attributes-of-phrase* records with one attribute: *freq-in-**B**=n*
- Sort the two phrase-tables: *freq-in-B* and *freq-in-C* and run the output through another "reducer" that
  - **appends** together all the attributes associated with the same key, so we now have elements like

Key	Value
`old man`	`freq(B)=10,freq(C)=13,phrasiness=740`
`bad service`	`freq(B)=8,freq(C)=25,phrasiness=254`
...	...

Generating and scoring phrases: 3

- Scan the through the phrase table one more time and add the informativeness attribute and the overall quality attribute

Key	Value
old man	freq(B)=10,freq(C)=13,informativeness=1.3,phrasiness=740
bad service	freq(B)=8,freq(C)=25,informativeness=560,phrasiness=254
...	...

Summary, assuming word vocabulary n_W is small:

- Scan foreground corpus C for phrases: $O(n_C)$ producing m_C phrase records – of course $m_C << n_C$
- Compute phrasiness: $O(m_C)$ Assumes word counts fit in memory
- Scan background corpus B for phrases: $O(n_B)$ producing m_B
- Sort together and combine records: $O(m \log m)$, $m=m_B + m_C$
- Compute informativeness and combined quality: $O(m)$

Ramping it up – keeping word counts out of memory

- Goal: records for *xy* with attributes *freq-in-B, freq-in-C, freq-of-x-in-C, freq-of-y-in-C, …*
- Assume I have built built phrase tables and word tables….how do I incorporate the word attributes into the phrase records?
- For each phrase *xy*, request necessary word frequencies:
  - Print "*x* ~request=freq-in-C,from=*xy*"
  - Print "*y* ~request=freq-in-C,from=*xy*"
- Sort all the word requests in with the word tables
- Scan through the result and generate the answers: for each word *w*, a_1=n_1,a_2=n_2,….
  - Print "*xy* ~request=freq-in-C,from=*w*"
- Sort the answers in with the *xy* records
- Scan through and augment the *xy* records appropriately

Generating and scoring phrases: 3

Summary

1. Scan foreground corpus C for phrases, words: $O(n_C)$ producing m_C phrase records, v_C word records
2. Scan phrase records producing word-freq requests: $O(m_C)$ producing $2m_C$ requests
3. Sort requests with word records: $O((2m_C + v_C)\log(2m_C + v_C)) = O(m_C \log m_C)$ since $v_C < m_C$
4. Scan through and answer requests: $O(m_C)$
5. Sort answers with phrase records: $O(m_C \log m_C)$
6. Repeat 1-5 for background corpus: $O(n_B + m_B \log m_B)$
7. Combine the two phrase tables: $O(m \log m)$, $m = m_B + m_C$
8. Compute all the statistics: $O(m)$

More cool work with phrases

- Turney: Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews.ACL '02.
- Task: review classification (65-85% accurate, depending on domain)
  - Identify candidate phrases (e.g., adj-noun bigrams, using POS tags)
  - Figure out the **semantic orientation** of each phrase using "pointwise mutual information" and aggregate

$$PMI(w_1, w_2) = log_2(p(w_1 \ and \ w_2)/p(w_1)p(w_2))$$

$$\mathrm{SO(phrase) = PMI(phrase,'excellent') - PMI(phrase,'poor')}$$

$$SO(phrase) = log_2(\frac{hits(phrase \ NEAR \ 'excellent')hits('excellent')}{hits(phrase \ NEAR \ 'poor')hits('poor')})$$

Table 2. An example of the processing of a review that the author has classified as *recommended*.[6]

Extracted Phrase	Part-of-Speech Tags	Semantic Orientation
online experience	JJ NN	2.253
low fees	JJ NNS	0.333
local branch	JJ NN	0.421
small part	JJ NN	0.053
online service	JJ NN	2.780
printable version	JJ NN	-0.705
direct deposit	JJ NN	1.288
well other	RB JJ	0.237
inconveniently located	RB VBN	-1.541
other bank	JJ NN	-0.850
true service	JJ NN	-0.732
Average Semantic Orientation		0.322

Table 3. An example of the processing of a review that the author has classified as *not recommended*.

Extracted Phrase	Part-of-Speech Tags	Semantic Orientation
little difference	JJ NN	-1.615
clever tricks	JJ NNS	-0.040
programs such	NNS JJ	0.117
possible moment	JJ NN	-0.668
unethical practices	JJ NNS	-8.484
low funds	JJ NNS	-6.843
old man	JJ NN	-2.566
other problems	JJ NNS	-2.748
probably wondering	RB VBG	-1.830
virtual monopoly	JJ NN	-2.050
other bank	JJ NN	-0.850
extra day	JJ NN	-0.286
direct deposits	JJ NNS	5.771
online web	JJ NN	1.936
cool thing	JJ NN	0.395
very handy	RB JJ	1.349
lesser evil	RBR JJ	-2.288
Average Semantic Orientation		-1.218

$$SO(phrase) = log_2\left(\frac{hits(phrase\ NEAR\ 'excellent')hits('excellent')}{hits(phrase\ NEAR\ 'poor')hits('poor')}\right)$$

“Answering Subcognitive Turing Test Questions: A Reply to French” - Turney

Robert French (1990, 2000) has argued that a disembodied computer cannot pass a Turing Test that includes *subcognitive* questions. He wrote (French, 2000):

> No computer that had not experienced the world as we humans had could pass a rigorously administered standard Turing Test. We show that the use of “subcognitive” questions allows the standard Turing Test to indirectly probe the human subcognitive associative concept network built up over a lifetime of experience with the world.

On a scale of 1 (awful) to 10 (excellent), please rate:

- How good is the name *Flugly* for a glamorous Hollywood actress?
- How good is the name *Flugly* for an accountant in a W.C. Fields movie?
- How good is the name *Flugly* for a child's teddy bear?

$$p(\text{Flu* } | \text{ actress}) = \frac{\text{hits((actress NEAR Flu*) AND glamorous)}}{\text{hits(actress AND glamorous)}}$$

LOW

$$p(\text{Flu* } | \text{ accountant}) = \frac{\text{hits((accountant NEAR Flu*) AND movie)}}{\text{hits(accountant AND movie)}}$$

HIGHER

$$p(\text{Flu* } | \text{ bear}) = \frac{\text{hits((bear NEAR Flu*) AND teddy)}}{\text{hits(bear AND teddy)}}$$

HIGHEST

On a scale of 1 (terrible) to 10 (excellent), please rate:

- banana peels as musical instruments
- coconut shells as musical instruments
- radios as musical instruments

Please rate the following smells (1 = very bad, 10 = very nice):

- Newly cut grass
- Freshly baked bread
- A wet bath towel
- The ocean
- A hospital corridor

On a scale of 1 (terrible) to 10 (excellent), please rate:

- banana peels as musical instruments
- coconut shells as musical instruments
- radios as musical instruments

Please rate the follo

- Newly cut grass
- Freshly baked b
- A wet bath towe
- The ocean
- A hospital corri

$p(\text{musical instruments} \mid \text{banana peels}) = 1 / 2{,}998 = 0.00033$

$p(\text{musical instruments} \mid \text{coconut shells}) = 5 / 1{,}880 = 0.0027$

$p(\text{musical instruments} \mid \text{radios}) = 1{,}253 / 1{,}006{,}207 = 0.0012$

$$p(\text{nice} \mid \text{newly cut grass}) = \frac{\text{hits}((\text{newly cut grass NEAR nice}) \text{ AND smell AND NOT } ((\text{newly cut grass OR nice}) \text{ NEAR "not"}))}{\text{hits}(\text{newly cut grass AND smell AND NOT } (\text{newly cut grass NEAR "not"}))}$$

Please rate the following smells (1 = very bad, 10 = very nice):

- Ne
- Fr
- A
- Th
- A

Table 2. Query results for questions about smell.

p(nice | newly cut grass)	= 1 / 102	= 0.0098
p(bad | newly cut grass)	= 0 / 102	= 0.0
p(nice | freshly baked bread)	= 8 / 848	= 0.0094
p(bad | freshly baked bread)	= 0 / 848	= 0.0
p(nice | wet bath towel)	= 0 / 3	= 0.0
p(bad | wet bath towel)	= 0 / 3	= 0.0
p(nice | ocean)	= 270 / 45,360	= 0.0060
p(bad | ocean)	= 107 / 45,360	= 0.0024
p(nice | hospital corridor)	= 0 / 134	= 0.0
p(bad | hospital corridor)	= 0 / 134	= 0.0

On a scale of 1 (terrible) to 10 (excellent), please rate:

- banana peels as musical instruments
- coconut shells as musical instruments
- radios as musical instruments

Please rate the following smells (1 = very bad, 10 = very nice):

- Newly cut grass
- Freshly baked bread
- A wet bath towel
- The ocean
- A hospital corridor

• Newly cut grass	= 10
• Freshly baked bread	= 10
• A wet bath towel	= 5
• The ocean	= 7
• A hospital corridor	= 5

More cool work with phrases

- Locating Complex Named Entities in Web Text. Doug Downey, Matthew Broadhead, and Oren Etzioni, IJCAI 2007.
- Task: identify complex named entities like "Proctor and Gamble", "War of 1812", "Dumb and Dumber", "Secretary of State William Cohen", …
- Formulation: decide whether to or not to merge nearby sequences of capitalized words *axb,* using variant of

$$c_k(a,b) = \frac{p(ab)^k}{p(a)p(b)} \Rightarrow g_k(a,b,c) = \frac{p(abc)^k}{p(a)p(b)p(c)}$$

- For k=1, ck is PM (w/o the log). For k=2, ck is "Symmetric Conditional Probability"

Downey et al results

	F1	Recall	Precision
SVMCMM	0.42	0.48	0.37
CRF	0.35	0.42	0.31
MAN	0.18	0.22	0.16
LEX	**0.63 (50%)**	**0.66**	**0.59**

Table 2: **Performance on Difficult Cases** LEX's F1 score is 50% higher than the nearest competitor, SVMCMM.

	F1	Recall	Precision
SVMCMM	0.29	0.34	0.25
CRF	0.25	0.31	0.21
MAN	0.18	0.22	0.16
LEX	**0.63 (117%)**	**0.66**	**0.60**

Table 4: **Performance on Unseen Entity Classes (Difficult Cases)** LEX outperforms its nearest competitor (SVMCMM) by 117%.

	F1	Recall	Precision
SVMCMM	0.96	0.96	0.96
CRF	0.94	0.94	0.95
MAN	0.97	0.96	0.98
LEX	0.97	0.97	0.97
CAPS	**1.00 (3%)**	**1.00**	**1.00**

Table 3: **Performance on Easy Cases** All methods perform comparably near the perfect performance of the CAPS baseline; CAPS outperforms LEX and MAN by 3%.

	F1	Recall	Precision
SVMCMM	0.93	0.92	0.94
CRF	0.94	0.93	0.95
MAN	0.97	0.96	0.98
LEX	0.95	0.95	0.95
CAPS	**1.00 (3%)**	**1.00**	**1.00**

Table 5: **Performance on Unseen Entity Classes (Easy Cases)** CAPS outperforms all methods by a small margin, performing 3% better than its nearest competitor (MAN).

	F1	Recall	Precision
LEX-PMI	0.38	0.43	0.34
LEX-SCP	**0.63 (66%)**	**0.66**	**0.59**

Table 1: **Performance of LEX using collocation measures PMI and SCP.** SCP outperforms PMI by 66% in F1.

Outline

- Even more on stream-and-sort and naïve Bayes
  - Request-answer pattern
- Another problem: “meaningful” phrase finding
  - Statistics for identifying phrases (or more generally correlations and differences)
  - Also using foreground and background corpora
- Implementing “phrase finding” efficiently
  - Using request-answer
- Some other phrase-related problems
  - Semantic orientation
  - Complex named entity recognition