Phrase Finding

William W. Cohen
Outline of the course

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

**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" 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" – 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” based on BLRT

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

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

<table>
<thead>
<tr>
<th></th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_1 )</td>
<td>( C(W_1=x \land W_2=y) ) how often bigram ( x \ y ) occurs in corpus ( C )</td>
</tr>
<tr>
<td>( n_1 )</td>
<td>( C(W_1=x) ) how often word ( x ) occurs in corpus ( C )</td>
</tr>
<tr>
<td>( k_2 )</td>
<td>( C(W_1 \neq x \land W_2=y) ) how often ( y ) occurs in ( C ) after a non-( x )</td>
</tr>
<tr>
<td>( n_2 )</td>
<td>( C(W_1 \neq x) ) how often a non-( x ) occurs in ( C )</td>
</tr>
</tbody>
</table>

Does \( y \) occur at the same frequency after \( x \) as in other positions?
“Informativeness” \(_1\) – based on BLRT

Define

\[ p_i = \frac{k_i}{n_i}, \quad p = \frac{(k_1 + k_2)}{(n_1 + n_2)}, \]

\[ L(p, k, n) = p^k (1-p)^{n-k} \]

\[ \phi_i(n_1, k_1, n_2, k_2) = 2 \log \frac{L(p_1, k_1, n_1)L(p_2, k_2, n_2)}{L(p, k_1, n_1)L(p, k_2, n_2)} \]

<table>
<thead>
<tr>
<th>(k_1)</th>
<th>(C(W_1=x \wedge W_2=y))</th>
<th>how often bigram (x\ y) occurs in corpus (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n_1)</td>
<td>(C(W_1=\ast \wedge W_2=\ast))</td>
<td>how many bigrams in corpus (C)</td>
</tr>
<tr>
<td>(k_2)</td>
<td>(B(W_1=x\wedge W_2=y))</td>
<td>how often (x\ y) occurs in background corpus</td>
</tr>
<tr>
<td>(n_2)</td>
<td>(B(W_1=\ast \wedge W_2=\ast))</td>
<td>how many bigrams in background corpus</td>
</tr>
</tbody>
</table>

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.

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

\[
\begin{align*}
\left(\log \frac{p}{1-p} = s\right) & \iff \left(p = \frac{1}{1+e^s}\right)
\end{align*}
\]

• Background corpus: 20 newsgroups dataset (20k messages, 7.4M words)
• Foreground corpus: rec.arts.movies.current-films June-Sep 2002 (4M words)
• Results?
<table>
<thead>
<tr>
<th></th>
<th>message news</th>
<th>16</th>
<th>sixth sense</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>minority report</td>
<td>17</td>
<td>hey kids</td>
</tr>
<tr>
<td>3</td>
<td>star wars</td>
<td>18</td>
<td>gaza man</td>
</tr>
<tr>
<td>4</td>
<td>john harkness</td>
<td>19</td>
<td>lee harrison</td>
</tr>
<tr>
<td>5</td>
<td>derek janssen</td>
<td>20</td>
<td>years ago</td>
</tr>
<tr>
<td>6</td>
<td>robert frenchu</td>
<td>21</td>
<td>julia roberts</td>
</tr>
<tr>
<td>7</td>
<td>sean o’hara</td>
<td>22</td>
<td>national guard</td>
</tr>
<tr>
<td>8</td>
<td>box office</td>
<td>23</td>
<td>bourne identity</td>
</tr>
<tr>
<td>9</td>
<td>dawn taylor</td>
<td>24</td>
<td>metrotoday <a href="http://www.zap2it.com">www.zap2it.com</a></td>
</tr>
<tr>
<td>10</td>
<td>anthony gaza</td>
<td>25</td>
<td>starweek magazine</td>
</tr>
<tr>
<td>11</td>
<td>star trek</td>
<td>26</td>
<td>eric chomko</td>
</tr>
<tr>
<td>12</td>
<td>ancient race</td>
<td>27</td>
<td>wilner starweek</td>
</tr>
<tr>
<td>13</td>
<td>scooby doo</td>
<td>28</td>
<td>tim gueguen</td>
</tr>
<tr>
<td>14</td>
<td>austin powers</td>
<td>29</td>
<td>jodie foster</td>
</tr>
<tr>
<td>15</td>
<td>home.attbi.com hey</td>
<td>30</td>
<td>johnnie kendricks</td>
</tr>
</tbody>
</table>
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”

$$\delta_w(p \ || \ q) \overset{\text{def}}{=} p(w) \log \frac{p(w)}{q(w)}$$
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”

\[ \delta_w(p \parallel q) \overset{\text{def}}{=} p(w) \log \frac{p(w)}{q(w)} \]

Phraseness: difference between bigram and unigram language model in foreground

\[ \delta_w(LM_{fg}^N \parallel LM_{fg}^1) \]

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"

$$\delta_w(p \parallel q) \overset{\text{def}}{=} p(w) \log \frac{p(w)}{q(w)}$$

Informativeness: difference between foreground and background models

$$\delta_w(LM_{fg}^N \parallel LM_{bg}^N), \text{ or}$$

$$\delta_w(LM_{fg}^1 \parallel LM_{bg}^1)$$

Bigram model: \( P(x y) = P(x)P(y \mid x) \)

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”

\[ \delta_w(p \parallel q) \overset{\text{def}}{=} p(w) \log \frac{p(w)}{q(w)} \]

Combined: difference between foreground bigram model and background unigram model

Bigram model: \( P(x y) = P(x)P(y \mid x) \)

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

\[ \delta_w(LM^{N}_{fg} \parallel LM^{1}_{bg}) \]
Pointwise KL, combined

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>old man</td>
<td>freq(B)=10, freq(C)=13, informativeness=1.3, phrasiness=740</td>
</tr>
<tr>
<td>bad service</td>
<td>freq(B)=8, freq(C)=25, informativeness=560, phrasiness=254</td>
</tr>
</tbody>
</table>
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: \( \text{freq-in-C} = n \)
- Stream through foreground corpus and count events “\( W_1 = x \)” in a (memory-based) hashtable….
- This is enough* to compute phrasiness:
  - \( \psi_p(x, y) = f(\text{freq-in-C}(x), \text{freq-in-C}(y), \text{freq-in-C}(x \ y)) \)
- …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 “$W_1=x \land W_2=y$” 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

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>old man</td>
<td>$freq(B)=10, freq(C)=13, phrasiness=740$</td>
</tr>
<tr>
<td>bad service</td>
<td>$freq(B)=8, freq(C)=25, phrasiness=254$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generating and scoring phrases: 3

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

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>old man</td>
<td>freq(B)=10, freq(C)=13, informativeness=1.3, phrasiness=740</td>
</tr>
<tr>
<td>bad service</td>
<td>freq(B)=8, freq(C)=25, informativeness=560, phrasiness=254</td>
</tr>
</tbody>
</table>

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 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 “$x \sim request=freq$-in-$C$, from=$xy$”
  – Print “$y \sim request=freq$-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 $w$, $a_1=n_1, a_2=n_2,\ldots$
  – Print “$xy \sim request=freq$-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

\[ PMI(w_1, w_2) = \log_2 \left( \frac{p(w_1 \text{ and } w_2)}{p(w_1)p(w_2)} \right) \]

\[ SO(\text{phrase}) = PMI(\text{phrase,'excellent'}) - PMI(\text{phrase,'poor'}) \]

\[ SO(\text{phrase}) = \log_2 \left( \frac{\text{hits}(\text{phrase NEAR 'excellent'}) \text{hits('excellent')}}{\text{hits}(\text{phrase NEAR 'poor'}) \text{hits('poor')}} \right) \]
Table 2. An example of the processing of a review that the author has classified as *recommended*.

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

Average Semantic Orientation: 0.322

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

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

Average Semantic Orientation: -1.218

\[
SO(\text{phrase}) = \log_2 \left( \frac{\text{hits}(\text{phrase NEAR 'excellent'})}{\text{hits('excellent')}} \frac{\text{hits('poor')}}{\text{hits}(\text{phrase NEAR 'poor'})} \right)
\]
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?

\[
p(\text{Flu}^{*} \mid \text{actress}) = \frac{\text{hits}((\text{actress NEAR Flu}^{*}) \text{ AND } \text{glamorous})}{\text{hits}(\text{actress AND } \text{glamorous})} \quad \text{LOW}
\]

\[
p(\text{Flu}^{*} \mid \text{accountant}) = \frac{\text{hits}((\text{accountant NEAR Flu}^{*}) \text{ AND } \text{movie})}{\text{hits}(\text{accountant AND movie})} \quad \text{HIGHER}
\]

\[
p(\text{Flu}^{*} \mid \text{bear}) = \frac{\text{hits}((\text{bear NEAR Flu}^{*}) \text{ AND } \text{teddy})}{\text{hits}(\text{bear AND teddy})} \quad \text{HIGHEST}
\]
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 following:

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

\[
\begin{align*}
p(\text{musical instruments} \mid \text{banana peels}) &= \frac{1}{2,998} = 0.00033 \\
p(\text{musical instruments} \mid \text{coconut shells}) &= \frac{5}{1,880} = 0.0027 \\
p(\text{musical instruments} \mid \text{radios}) &= \frac{1,253}{1,006,207} = 0.0012
\end{align*}
\]
\[ p(\text{nice} \mid \text{newly cut grass}) = \]

\[ \frac{\text{hits}((\text{newly cut grass NEAR nice}) \text{ AND smell AND NOT ((newly cut grass OR nice) NEAR "not"))}}{\text{hits(newly cut grass AND smell AND NOT (newly cut grass NEAR "not"))}} \]

Please rate the following smells (1 = worst, 10 = best):

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

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

<table>
<thead>
<tr>
<th>Smell</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newly cut grass</td>
<td>10</td>
</tr>
<tr>
<td>Freshly baked bread</td>
<td>10</td>
</tr>
<tr>
<td>A wet bath towel</td>
<td>5</td>
</tr>
<tr>
<td>The ocean</td>
<td>7</td>
</tr>
<tr>
<td>A hospital corridor</td>
<td>5</td>
</tr>
</tbody>
</table>
More cool work with phrases

• 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

\[
c_k(a, b) = \frac{p(ab)^k}{p(a)p(b)} \quad \rightarrow \quad g_k(a, b, c) = \frac{p(abc)^k}{p(a)p(b)p(c)}
\]

• For $k=1$, $c_k$ is PM (w/o the log). For $k=2$, $c_k$ is “Symmetric Conditional Probability”
### Downey et al results

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMCMM</td>
<td>0.42</td>
<td>0.48</td>
<td>0.37</td>
</tr>
<tr>
<td>CRF</td>
<td>0.35</td>
<td>0.42</td>
<td>0.31</td>
</tr>
<tr>
<td>MAN</td>
<td>0.18</td>
<td>0.22</td>
<td>0.16</td>
</tr>
<tr>
<td>LEX</td>
<td><strong>0.63 (50%)</strong></td>
<td><strong>0.66</strong></td>
<td><strong>0.59</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMCMM</td>
<td>0.29</td>
<td>0.34</td>
<td>0.25</td>
</tr>
<tr>
<td>CRF</td>
<td>0.25</td>
<td>0.31</td>
<td>0.21</td>
</tr>
<tr>
<td>MAN</td>
<td>0.18</td>
<td>0.22</td>
<td>0.16</td>
</tr>
<tr>
<td>LEX</td>
<td><strong>0.63 (117%)</strong></td>
<td><strong>0.66</strong></td>
<td><strong>0.60</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMCMM</td>
<td>0.96</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>CRF</td>
<td>0.94</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>MAN</td>
<td>0.97</td>
<td>0.96</td>
<td>0.98</td>
</tr>
<tr>
<td>LEX</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
</tr>
<tr>
<td>CAPS</td>
<td><strong>1.00 (3%)</strong></td>
<td><strong>1.00</strong></td>
<td><strong>1.00</strong></td>
</tr>
</tbody>
</table>

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%.

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMCMM</td>
<td>0.93</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>CRF</td>
<td>0.94</td>
<td>0.93</td>
<td>0.95</td>
</tr>
<tr>
<td>MAN</td>
<td>0.97</td>
<td>0.96</td>
<td>0.98</td>
</tr>
<tr>
<td>LEX</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>CAPS</td>
<td><strong>1.00 (3%)</strong></td>
<td><strong>1.00</strong></td>
<td><strong>1.00</strong></td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEX-PMI</td>
<td>0.38</td>
<td>0.43</td>
<td>0.34</td>
</tr>
<tr>
<td>LEX-SCP</td>
<td><strong>0.63 (66%)</strong></td>
<td><strong>0.66</strong></td>
<td><strong>0.59</strong></td>
</tr>
</tbody>
</table>

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