Search Engines
Chapter 4 – Processing Text

7.5.2009
Felix Naumann
Converting documents to *index terms*

- "Text processing" or "Text transformation"

Easy: Do nothing

- Why?
  - Matching the exact string of characters typed by the user is too restrictive.
    - Poor effectiveness
  - Not all words are of equal value in a search
  - Sometimes not clear where words begin and end
    - Not even clear what a word is in some languages
      - e.g., Chinese, Korean
Processing Text

- NLP
  - Syntactic analysis
  - Semantic analysis
- Text statistics
  - Counting words
  - Counting co-occurrences
- Many simple techniques
  - Lower case
  - Punctuation
  - Tokenization
  - Stopping
  - Stemming
  - Structure and format
  - Links
- But profound impact
Overview

- Text statistics
- Document parsing
- Link Analysis
- Information Extraction
Huge variety of words used in text but...

Many statistical characteristics of word occurrences are predictable
  - e.g., distribution of word counts

Retrieval models and ranking algorithms depend heavily on statistical properties of words.
  - e.g., important words occur often in a document but are not of high frequency in entire collection
  - tf-idf
Zipf’s Law

- Distribution of word frequencies is very skewed.
  - A few words occur very often, many words hardly ever occur
  - Two most common words ("the", "of") make up about 10% of all word occurrences in text documents
  - Top 6 words account for 20% of text.
  - Top 50 words account for 40% of text.
  - And: 50% of all words in a large sample occur only once.

- Zipf’s “law”:
  - Observation that rank \( r \) of a word times its frequency \( f \) is approximately a constant \( k \)
    - Assuming words are ranked in order of decreasing frequency
  - \( r \cdot f \approx k \) or \( r \cdot P_r \approx c \)
    - where \( P_r \) is probability of word occurrence and \( c \approx 0.1 \) for English
Zipf’s Law

George Kingsley Zipf (1902–1950)

\[ r \cdot P_r = c \text{ for } c = 0.1 \]

\[ \Leftrightarrow P_r = 0.1 / r \]

Probability (of occurrence)

Rank (by decreasing frequency)

http://www.lib.jgypk.u-szeged.hu/alknyelv/idegenek/klasszikusok/Zipf/nyelvlesz.htm
News Collection (AP89) Statistics

- Associated Press from 1989

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total documents</td>
<td>84,678</td>
</tr>
<tr>
<td>Total word occurrences</td>
<td>39,749,179</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>198,763</td>
</tr>
<tr>
<td>Words occurring &gt; 1000 times</td>
<td>4,169</td>
</tr>
<tr>
<td>Words occurring once</td>
<td>70,064</td>
</tr>
</tbody>
</table>
# Top 50 Words from AP89

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq.</th>
<th>$r$.</th>
<th>$P_r$ (%)</th>
<th>$r.P_r$</th>
<th>Word</th>
<th>Freq.</th>
<th>$r$.</th>
<th>$P_r$ (%)</th>
<th>$r.P_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2,420,778</td>
<td>1</td>
<td>6.49</td>
<td>0.065</td>
<td>has</td>
<td>136,007</td>
<td>26</td>
<td>0.37</td>
<td>0.095</td>
</tr>
<tr>
<td>of</td>
<td>1,045,733</td>
<td>2</td>
<td>2.80</td>
<td>0.056</td>
<td>are</td>
<td>130,322</td>
<td>27</td>
<td>0.35</td>
<td>0.094</td>
</tr>
<tr>
<td>to</td>
<td>968,882</td>
<td>3</td>
<td>2.60</td>
<td>0.078</td>
<td>not</td>
<td>127,493</td>
<td>28</td>
<td>0.34</td>
<td>0.096</td>
</tr>
<tr>
<td>a</td>
<td>892,429</td>
<td>4</td>
<td>2.39</td>
<td>0.096</td>
<td>who</td>
<td>116,364</td>
<td>29</td>
<td>0.31</td>
<td>0.090</td>
</tr>
<tr>
<td>and</td>
<td>865,644</td>
<td>5</td>
<td>2.32</td>
<td>0.120</td>
<td>they</td>
<td>111,024</td>
<td>30</td>
<td>0.30</td>
<td>0.089</td>
</tr>
<tr>
<td>in</td>
<td>847,825</td>
<td>6</td>
<td>2.27</td>
<td>0.140</td>
<td>its</td>
<td>111,021</td>
<td>31</td>
<td>0.30</td>
<td>0.092</td>
</tr>
<tr>
<td>said</td>
<td>504,593</td>
<td>7</td>
<td>1.35</td>
<td>0.095</td>
<td>had</td>
<td>103,943</td>
<td>32</td>
<td>0.28</td>
<td>0.089</td>
</tr>
<tr>
<td>for</td>
<td>363,865</td>
<td>8</td>
<td>0.98</td>
<td>0.078</td>
<td>will</td>
<td>102,949</td>
<td>33</td>
<td>0.28</td>
<td>0.091</td>
</tr>
<tr>
<td>that</td>
<td>347,072</td>
<td>9</td>
<td>0.93</td>
<td>0.084</td>
<td>would</td>
<td>99,503</td>
<td>34</td>
<td>0.27</td>
<td>0.091</td>
</tr>
<tr>
<td>was</td>
<td>293,027</td>
<td>10</td>
<td>0.79</td>
<td>0.099</td>
<td>about</td>
<td>92,983</td>
<td>35</td>
<td>0.25</td>
<td>0.089</td>
</tr>
<tr>
<td>on</td>
<td>291,947</td>
<td>11</td>
<td>0.78</td>
<td>0.086</td>
<td>i</td>
<td>92,005</td>
<td>36</td>
<td>0.25</td>
<td>0.089</td>
</tr>
<tr>
<td>he</td>
<td>250,919</td>
<td>12</td>
<td>0.67</td>
<td>0.081</td>
<td>been</td>
<td>88,786</td>
<td>37</td>
<td>0.24</td>
<td>0.088</td>
</tr>
<tr>
<td>is</td>
<td>245,843</td>
<td>13</td>
<td>0.65</td>
<td>0.086</td>
<td>this</td>
<td>87,286</td>
<td>38</td>
<td>0.23</td>
<td>0.089</td>
</tr>
<tr>
<td>with</td>
<td>223,846</td>
<td>14</td>
<td>0.60</td>
<td>0.084</td>
<td>their</td>
<td>84,638</td>
<td>39</td>
<td>0.23</td>
<td>0.089</td>
</tr>
<tr>
<td>at</td>
<td>210,064</td>
<td>15</td>
<td>0.56</td>
<td>0.085</td>
<td>new</td>
<td>83,449</td>
<td>40</td>
<td>0.22</td>
<td>0.090</td>
</tr>
<tr>
<td>by</td>
<td>209,586</td>
<td>16</td>
<td>0.56</td>
<td>0.090</td>
<td>or</td>
<td>81,796</td>
<td>41</td>
<td>0.22</td>
<td>0.090</td>
</tr>
<tr>
<td>it</td>
<td>195,621</td>
<td>17</td>
<td>0.52</td>
<td>0.089</td>
<td>which</td>
<td>80,385</td>
<td>42</td>
<td>0.22</td>
<td>0.091</td>
</tr>
<tr>
<td>from</td>
<td>189,451</td>
<td>18</td>
<td>0.51</td>
<td>0.091</td>
<td>we</td>
<td>80,245</td>
<td>43</td>
<td>0.22</td>
<td>0.093</td>
</tr>
<tr>
<td>as</td>
<td>181,714</td>
<td>19</td>
<td>0.49</td>
<td>0.093</td>
<td>more</td>
<td>76,388</td>
<td>44</td>
<td>0.21</td>
<td>0.090</td>
</tr>
<tr>
<td>be</td>
<td>157,300</td>
<td>20</td>
<td>0.42</td>
<td>0.084</td>
<td>after</td>
<td>75,165</td>
<td>45</td>
<td>0.20</td>
<td>0.091</td>
</tr>
<tr>
<td>were</td>
<td>153,913</td>
<td>21</td>
<td>0.41</td>
<td>0.087</td>
<td>us</td>
<td>72,045</td>
<td>46</td>
<td>0.19</td>
<td>0.089</td>
</tr>
<tr>
<td>an</td>
<td>152,576</td>
<td>22</td>
<td>0.41</td>
<td>0.090</td>
<td>percent</td>
<td>71,956</td>
<td>47</td>
<td>0.19</td>
<td>0.091</td>
</tr>
<tr>
<td>have</td>
<td>149,749</td>
<td>23</td>
<td>0.40</td>
<td>0.092</td>
<td>up</td>
<td>71,082</td>
<td>48</td>
<td>0.19</td>
<td>0.092</td>
</tr>
<tr>
<td>his</td>
<td>142,285</td>
<td>24</td>
<td>0.38</td>
<td>0.092</td>
<td>one</td>
<td>70,266</td>
<td>49</td>
<td>0.19</td>
<td>0.092</td>
</tr>
<tr>
<td>but</td>
<td>140,880</td>
<td>25</td>
<td>0.38</td>
<td>0.094</td>
<td>people</td>
<td>68,988</td>
<td>50</td>
<td>0.19</td>
<td>0.093</td>
</tr>
</tbody>
</table>

$r.P_r$ value always close to 0.1
Zipf is most inaccurate for very frequent and very infrequent words.

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq.</th>
<th>r</th>
<th>$P_r$ (%)</th>
<th>$r.P_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2,420,778</td>
<td>1</td>
<td>6.49</td>
<td>0.065</td>
</tr>
<tr>
<td>of</td>
<td>1,045,733</td>
<td>2</td>
<td>2.80</td>
<td>0.056</td>
</tr>
<tr>
<td>to</td>
<td>968,882</td>
<td>3</td>
<td>2.60</td>
<td>0.078</td>
</tr>
<tr>
<td>a</td>
<td>892,429</td>
<td>4</td>
<td>2.39</td>
<td>0.096</td>
</tr>
<tr>
<td>and</td>
<td>865,644</td>
<td>5</td>
<td>2.32</td>
<td>0.120</td>
</tr>
<tr>
<td>in</td>
<td>847,825</td>
<td>6</td>
<td>2.27</td>
<td>0.140</td>
</tr>
<tr>
<td>said</td>
<td>504,593</td>
<td>7</td>
<td>1.35</td>
<td>0.095</td>
</tr>
<tr>
<td>for</td>
<td>363,865</td>
<td>8</td>
<td>0.98</td>
<td>0.078</td>
</tr>
<tr>
<td>that</td>
<td>347,072</td>
<td>9</td>
<td>0.93</td>
<td>0.084</td>
</tr>
<tr>
<td>was</td>
<td>293,027</td>
<td>10</td>
<td>0.79</td>
<td>0.079</td>
</tr>
<tr>
<td>on</td>
<td>291,947</td>
<td>11</td>
<td>0.78</td>
<td>0.086</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Word</th>
<th>Freq.</th>
<th>r</th>
<th>$P_r$ (%)</th>
<th>$r.P_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assistant</td>
<td>5,095</td>
<td>1,021</td>
<td>.013</td>
<td>0.13</td>
</tr>
<tr>
<td>Sewers</td>
<td>100</td>
<td>17,110</td>
<td>$2.56 \times 10^{-4}$</td>
<td>0.04</td>
</tr>
<tr>
<td>Toothbrush</td>
<td>10</td>
<td>51,555</td>
<td>$2.56 \times 10^{-5}$</td>
<td>0.01</td>
</tr>
<tr>
<td>Hazmat</td>
<td>1</td>
<td>166,945</td>
<td>$2.56 \times 10^{-6}$</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Zipf’s Law for AP89

Note problems at high and low frequencies
Zipf’s Law

- Reminder: \( r \cdot f \approx k \)

- What is the proportion of words with a given frequency?
  - Word that occurs \( n \) times has rank \( r_n = k/n \)
  - Multiple words can have same frequency
    - \( r_n \) is associated with last word in group
  - Number of words with same frequency \( n \) is
    - \( r_n - r_{n+1} = k/n - k/(n + 1) = k/n(n + 1) \)
  - Proportion found by dividing by total number of words
    - = rank of last word with freq. 1 = highest rank = \( k/1 = k \)
  - So, proportion with frequency \( n \) is \( 1/n(n+1) \)
    - = half of all words appear once
      - \( (n=1 => \text{proportion} = 1/2) \)
Zipf’s Law

- Example word frequency ranking

<table>
<thead>
<tr>
<th>Rank</th>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>concern</td>
<td>5,100</td>
</tr>
<tr>
<td>1001</td>
<td>spoke</td>
<td>5,100</td>
</tr>
<tr>
<td>1002</td>
<td>summit</td>
<td>5,100</td>
</tr>
<tr>
<td>1003</td>
<td>bring</td>
<td>5,099</td>
</tr>
<tr>
<td>1004</td>
<td>star</td>
<td>5,099</td>
</tr>
<tr>
<td>1005</td>
<td>immediate</td>
<td>5,099</td>
</tr>
<tr>
<td>1006</td>
<td>chemical</td>
<td>5,099</td>
</tr>
<tr>
<td>1007</td>
<td>african</td>
<td>5,098</td>
</tr>
</tbody>
</table>

- To compute number of words with frequency 5,099
  - rank of “chemical” minus the rank of “summit”
  - $1006 - 1002 = 4$

- Proportion: $\frac{1}{n(n+1)} = \frac{1}{5,099 \times 5,100} = \frac{1}{26,004,900}$
### Proportions of words occurring n times in 336,310 TREC documents

<table>
<thead>
<tr>
<th>Number of Occurrences (n)</th>
<th>Predicted Proportion (1/n(n+1))</th>
<th>Actual Proportion</th>
<th>Actual Number of Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.500</td>
<td>.402</td>
<td>204,357</td>
</tr>
<tr>
<td>2</td>
<td>.167</td>
<td>.132</td>
<td>67,082</td>
</tr>
<tr>
<td>3</td>
<td>.083</td>
<td>.069</td>
<td>35,083</td>
</tr>
<tr>
<td>4</td>
<td>.050</td>
<td>.046</td>
<td>23,271</td>
</tr>
<tr>
<td>5</td>
<td>.033</td>
<td>.032</td>
<td>16,332</td>
</tr>
<tr>
<td>6</td>
<td>.024</td>
<td>.024</td>
<td>12,421</td>
</tr>
<tr>
<td>7</td>
<td>.018</td>
<td>.019</td>
<td>9,766</td>
</tr>
<tr>
<td>8</td>
<td>.014</td>
<td>.016</td>
<td>8,200</td>
</tr>
<tr>
<td>9</td>
<td>.011</td>
<td>.014</td>
<td>6,907</td>
</tr>
<tr>
<td>10</td>
<td>.009</td>
<td>.012</td>
<td>5,893</td>
</tr>
</tbody>
</table>

- Vocabulary size is 508,209
Vocabulary Growth

- As corpus grows, so does vocabulary size
  - But: Fewer new words when corpus is already large

Observed relationship (*Heaps’ Law, found empirically*):

\[ v = k \cdot n^\beta \]

- where \( v \) is vocabulary size (number of unique words)
- \( n \) is the number of words in corpus (non-unique)
- \( k, \beta \) are parameters that vary for each corpus
  - typical values given are \( 10 \leq k \leq 100 \) and \( \beta \approx 0.5 \)

- Example
  - \( n = 1,000,000 \quad k = 50 \quad \beta = 0.5 \)
  - \( v = 50 \cdot 1,000,000^{0.5} = 50,000 \)
Heaps law with $\beta = 0.455$ and $k = 62.95$
Heaps’ Law Predictions

- Predictions for TREC collections are accurate for large numbers of words
  - e.g., first 10,879,522 words of the AP89 collection scanned
  - prediction is 100,151 unique words
  - actual number is 100,024
- Predictions for small numbers of words (i.e. < 1,000) are much worse
25 billion, and still many new words
Heaps’ Law works with very large corpora
- New words occurring even after seeing 30 million!
- Parameter values on Web different than typical TREC values

New words come from a variety of sources
- spelling errors, invented words (e.g. product, company names), code, other languages, email addresses, etc.

Search engines must deal with these large and growing vocabularies
Estimating Result Set Size

- How many pages contain all of the query terms?
  - Not always conjunctive semantics

For the query “a b c”:

\[ f_{abc} = \frac{N \cdot f_a}{N} \cdot \frac{f_b}{N} \cdot \frac{f_c}{N} = \frac{(f_a \cdot f_b \cdot f_c)}{N^2} \]

- Assuming that terms occur independently
- \( f_{abc} \) is the estimated size of the result set
- \( f_a, f_b, f_c \) are the number of documents that terms a, b, and c occur in
  - Available through index
  - Document frequency (not word occurrences)
- \( N \) is the number of documents in the collection
TREC GOV2 Example

Collection size \((N)\) is 25,205,179

<table>
<thead>
<tr>
<th>Word(s)</th>
<th>Document Frequency</th>
<th>Estimated Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>tropical</td>
<td>120,990</td>
<td></td>
</tr>
<tr>
<td>fish</td>
<td>1,131,855</td>
<td></td>
</tr>
<tr>
<td>aquarium</td>
<td>26,480</td>
<td></td>
</tr>
<tr>
<td>breeding</td>
<td>81,885</td>
<td></td>
</tr>
<tr>
<td>tropical fish</td>
<td>18,472</td>
<td>5,433</td>
</tr>
<tr>
<td>tropical aquarium</td>
<td>1,921</td>
<td>127</td>
</tr>
<tr>
<td>tropical breeding</td>
<td>5,510</td>
<td>393</td>
</tr>
<tr>
<td>fish aquarium</td>
<td>9,722</td>
<td>1,189</td>
</tr>
<tr>
<td>fish breeding</td>
<td>36,427</td>
<td>3,677</td>
</tr>
<tr>
<td>aquarium breeding</td>
<td>1,848</td>
<td>86</td>
</tr>
<tr>
<td>tropical fish aquarium</td>
<td>1,529</td>
<td>6</td>
</tr>
<tr>
<td>tropical fish breeding</td>
<td>3,629</td>
<td>18</td>
</tr>
</tbody>
</table>
Result Set Size Estimation

- Poor estimates because words are not independent
- Better estimates possible if pair-wise co-occurrence information is available
  - $P(a \cap b \cap c) = P(a \cap b) \cdot P(c|(a \cap b))$
  - Approximate $P(c|(a \cap b))$ with $\max[P(c|a), P(c|b)]$
  - $P(c|a) = P(c \cap a)/P(a)$
  - $f_{tropical \cap fish \cap aquarium} = f_{tropical \cap aquarium} \cdot f_{fish \cap aquarium}/f_{aquarium} = 1921 \cdot 9722/26480 = 705$
  - $f_{tropical \cap fish \cap breeding} = f_{tropical \cap breeding} \cdot f_{fish \cap breeding}/f_{breeding} = 5510 \cdot 36427/81885 = 2451$
- Still too low, because still some independence assumptions
- Storing deeper co-occurrence (triples, quadruples, ...) too expensive
Even better estimates using initial result set during processing

- Estimate is simply $C/s$
  - $s$ is the proportion of the total documents that have been ranked
  - $C$ is the number of documents found that contain all the query words
- E.g., “tropical fish aquarium” in GOV2
  - after processing 3,000 out of the 26,480 documents that contain “aquarium”, $C = 258$
    \[
    f_{tropical \cap fish \cap aquarium} = \frac{258}{(3000 \div 26480)} = 2,277
    \]
    \[
    ( = 26480 \cdot \frac{258}{3000} )
    \]
  - After processing 20% of the documents
    \[
    f_{tropical \cap fish \cap aquarium} = 1,778 \quad (1,529 \text{ is real value})
    \]
- Total number of documents in collection irrelevant here
Important issue for Web search engines

- Academia: How big is the web?
- Business: Which search engine has best coverage?

Simple technique: Use independence model

- Given two words $a$ and $b$ that are (probably) independent

\[
\frac{f_{ab}}{N} = \frac{f_a}{N} \cdot \frac{f_b}{N}
\]

\[
N = \frac{f_a \cdot f_b}{f_{ab}}
\]

- e.g., for GOV2

\[
\begin{align*}
    f_{\text{lincoln}} &= 771,326 \\
    f_{\text{tropical}} &= 120,990 \\
    f_{\text{lincoln} \cap \text{tropical}} &= 3,018 \\
    N &= (120990 \cdot 771326)/3018 = 30,922,045 \\
    \text{(actual number is 25,205,179)}
\end{align*}
\]
Estimating Google’s Size (GS)

GS = \((126,000,000 \cdot 79,900,000) / 2,740,000 = 3,674,233,577\)

Actual size: 1,000,000,000,000
Overview

- Text statistics
- Document parsing
- Link Analysis
- Information Extraction
Motivation

- Document parsing = Recognition of content and structure of document
- Tokenizing / lexical analysis = recognizing words in sequence of characters
- Syntactic analysis = recognizing structure for content
- Parsing very tolerant – represent every document in index!

- Input: Result of crawling – textual representation of web page
  - With markup
- Output: Data structure used for index
Tokenizing

- Forming words from sequence of characters
- Surprisingly complex in English, can be harder in other languages
- Early IR systems:
  - Any sequence of alphanumeric characters of length > 3
  - Terminated by a space or other special character
  - Any upper-case changed to lower-case (case-folding, downcasing)
- Example:
  - “Bigcorp's 2007 bi-annual report showed profits rose 10%.”
  - becomes “bigcorp 2007 annual report showed profits rose”
- Too simple for search applications or even large-scale experiments
- Why? Too much information lost
  - Small decisions in tokenizing can have major impact on effectiveness of some queries
Small words can be important in some queries, usually in combinations.

- \( xp, ma, pm, ben\ e \ king, el\ paso, system\ r \)
- \( master\ p, gm, j\ lo, world\ war\ II \)

Both hyphenated and non-hyphenated forms of many words are common.

- Sometimes hyphen is not needed
  - \( e\-bay, wal\-mart, active\-x, cd\-rom, t\-shirts \)
- At other times, hyphens should be considered either as part of the word or a word separator
  - \( winston\-salem, mazda\ rx\-7, e\-cards, pre\-diabetes, t\-mobile, spanish\-speaking \)
Tokenizing Problems

- Special characters are an important part of tags, URLs, code in documents
- Capitalized words can have different meaning from lower case words
  - *Bush, Apple*
  - *bush, apple*
- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - *rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner's*
Die Kapostroph-Gruselgalerie – Kategorie „Völlig willenlos“
http://www.apostroph.de/

Motto: "Bei mir ist nicht's für die Katz!"

Nicht's wie weg hier

Bitte die Tür zum Hof steht's verschlossen halten!!!
Tokenizing Problems

- Numbers can be important, including decimals
  - `nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358`

- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - `I.B.M., Ph.D., cs.umass.edu, F.E.A.R.`

- Note: tokenizing steps for queries must be identical to steps for documents
## Tokenizing Process

- **Step 1: Parse for markup**
  - Allow for syntax errors
  - Identify appropriate parts of document to tokenize

- **Step 2: Parse for content**
  - Defer complex decisions to other components
    - Stemming, dates, NER
  - Word is any sequence of alphanumeric characters, terminated by a space or special character, with everything converted to lower-case
    - Let query transformation component deal with ambiguities
  - Example: 92.3 → 92 3 but search finds documents with 92 and 3 adjacent
  - Incorporate some rules to reduce dependence on query transformation components
Tokenizing Process

- Not that different than simple tokenizing process used in past
- Examples of rules used with TREC
  - Apostrophes in words ignored
    - o’connor → oconnor  bob’s → bobs
  - Periods in abbreviations ignored
    - I.B.M. → ibm  Ph.D. → ph d
Stopping

- Function words (determiners, prepositions) have little meaning on their own
  - Determiners: The, a, an, that, those, ...
  - Prepositions: Over, under, above, below, ...
- High occurrence frequencies
- Little relevance (except for phrases)
- Treated as *stopwords* (i.e., removed)
  - Reduce index space,
  - improve response time
  - improve effectiveness
- Can be important in combinations
  - e.g., “to be or not to be”

| Word | Freq. | r | $P_r$ (%) | $r.P_r$ | Word | Freq. | r | $P_r$ (%) | $r.P_r$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>2,420,778</td>
<td>1</td>
<td>6.49</td>
<td>0.065</td>
<td>has</td>
<td>136,007</td>
<td>26</td>
<td>0.37</td>
<td>0.095</td>
</tr>
<tr>
<td>of</td>
<td>1,045,733</td>
<td>2</td>
<td>2.80</td>
<td>0.056</td>
<td>are</td>
<td>130,322</td>
<td>27</td>
<td>0.35</td>
<td>0.094</td>
</tr>
<tr>
<td>to</td>
<td>968,882</td>
<td>3</td>
<td>2.60</td>
<td>0.078</td>
<td>not</td>
<td>127,493</td>
<td>28</td>
<td>0.34</td>
<td>0.096</td>
</tr>
<tr>
<td>a</td>
<td>892,429</td>
<td>4</td>
<td>2.39</td>
<td>0.096</td>
<td>who</td>
<td>116,364</td>
<td>29</td>
<td>0.31</td>
<td>0.090</td>
</tr>
<tr>
<td>and</td>
<td>865,644</td>
<td>5</td>
<td>2.32</td>
<td>0.120</td>
<td>they</td>
<td>111,024</td>
<td>30</td>
<td>0.30</td>
<td>0.089</td>
</tr>
<tr>
<td>in</td>
<td>847,825</td>
<td>6</td>
<td>2.27</td>
<td>0.140</td>
<td>its</td>
<td>111,021</td>
<td>31</td>
<td>0.30</td>
<td>0.092</td>
</tr>
<tr>
<td>said</td>
<td>504,593</td>
<td>7</td>
<td>1.35</td>
<td>0.095</td>
<td>had</td>
<td>103,943</td>
<td>32</td>
<td>0.28</td>
<td>0.089</td>
</tr>
<tr>
<td>for</td>
<td>363,865</td>
<td>8</td>
<td>0.98</td>
<td>0.076</td>
<td>will</td>
<td>102,949</td>
<td>33</td>
<td>0.28</td>
<td>0.091</td>
</tr>
<tr>
<td>that</td>
<td>347,072</td>
<td>9</td>
<td>0.93</td>
<td>0.084</td>
<td>would</td>
<td>99,503</td>
<td>34</td>
<td>0.27</td>
<td>0.091</td>
</tr>
<tr>
<td>was</td>
<td>293,027</td>
<td>10</td>
<td>0.79</td>
<td>0.079</td>
<td>about</td>
<td>92,983</td>
<td>35</td>
<td>0.25</td>
<td>0.087</td>
</tr>
<tr>
<td>on</td>
<td>219,947</td>
<td>11</td>
<td>0.78</td>
<td>0.086</td>
<td>i</td>
<td>92,005</td>
<td>36</td>
<td>0.25</td>
<td>0.089</td>
</tr>
<tr>
<td>he</td>
<td>250,919</td>
<td>12</td>
<td>0.67</td>
<td>0.081</td>
<td>been</td>
<td>88,786</td>
<td>37</td>
<td>0.24</td>
<td>0.088</td>
</tr>
<tr>
<td>is</td>
<td>245,843</td>
<td>13</td>
<td>0.65</td>
<td>0.084</td>
<td>this</td>
<td>87,286</td>
<td>38</td>
<td>0.23</td>
<td>0.089</td>
</tr>
<tr>
<td>with</td>
<td>223,846</td>
<td>14</td>
<td>0.60</td>
<td>0.084</td>
<td>their</td>
<td>84,638</td>
<td>39</td>
<td>0.23</td>
<td>0.089</td>
</tr>
<tr>
<td>at</td>
<td>210,064</td>
<td>15</td>
<td>0.56</td>
<td>0.085</td>
<td>new</td>
<td>83,449</td>
<td>40</td>
<td>0.22</td>
<td>0.090</td>
</tr>
<tr>
<td>by</td>
<td>209,586</td>
<td>16</td>
<td>0.56</td>
<td>0.090</td>
<td>or</td>
<td>81,796</td>
<td>41</td>
<td>0.22</td>
<td>0.090</td>
</tr>
<tr>
<td>it</td>
<td>195,621</td>
<td>17</td>
<td>0.52</td>
<td>0.089</td>
<td>which</td>
<td>80,385</td>
<td>42</td>
<td>0.22</td>
<td>0.091</td>
</tr>
<tr>
<td>from</td>
<td>189,451</td>
<td>18</td>
<td>0.51</td>
<td>0.091</td>
<td>we</td>
<td>80,245</td>
<td>43</td>
<td>0.22</td>
<td>0.093</td>
</tr>
<tr>
<td>as</td>
<td>181,714</td>
<td>19</td>
<td>0.49</td>
<td>0.093</td>
<td>more</td>
<td>76,388</td>
<td>44</td>
<td>0.21</td>
<td>0.090</td>
</tr>
<tr>
<td>be</td>
<td>157,300</td>
<td>20</td>
<td>0.42</td>
<td>0.084</td>
<td>after</td>
<td>75,165</td>
<td>45</td>
<td>0.20</td>
<td>0.091</td>
</tr>
<tr>
<td>were</td>
<td>153,913</td>
<td>21</td>
<td>0.41</td>
<td>0.087</td>
<td>us</td>
<td>72,045</td>
<td>46</td>
<td>0.19</td>
<td>0.089</td>
</tr>
<tr>
<td>an</td>
<td>152,576</td>
<td>22</td>
<td>0.41</td>
<td>0.090</td>
<td>percent</td>
<td>71,956</td>
<td>47</td>
<td>0.19</td>
<td>0.091</td>
</tr>
<tr>
<td>have</td>
<td>149,749</td>
<td>23</td>
<td>0.40</td>
<td>0.092</td>
<td>up</td>
<td>71,082</td>
<td>48</td>
<td>0.19</td>
<td>0.092</td>
</tr>
<tr>
<td>his</td>
<td>142,285</td>
<td>24</td>
<td>0.38</td>
<td>0.092</td>
<td>one</td>
<td>70,266</td>
<td>49</td>
<td>0.19</td>
<td>0.092</td>
</tr>
<tr>
<td>but</td>
<td>140,880</td>
<td>25</td>
<td>0.38</td>
<td>0.094</td>
<td>people</td>
<td>68,988</td>
<td>50</td>
<td>0.19</td>
<td>0.093</td>
</tr>
</tbody>
</table>

Felix Naumann | Search Engines | Sommer 2009
Stopping

- Stopword list can be created from high-frequency words or based on a standard list
  - With caution
- Lists are customized for applications, domains, and even parts of documents
  - e.g., “click” is a good stopword for anchor text
- Best policy is to index all words in documents, make decisions about which words to use at query time
  - Stopwords are removed from query, except with “+”-sign
  - But: Space consuming
Stemming

- Also: Conflation
- Many morphological variations of words
  - *inflectional* (plurals, tenses)
    - Flexion, Beugung: Kasus, Numerus, Genus, Tempus
  - *derivational* (making verbs nouns etc.)
    - Ableitung und Zusammensetzung (Komposition)
- In most cases, these have the same or very similar meanings
- Stemmers attempt to reduce morphological variations of words to a common stem
  - Usually involves removing suffixes
- Can be done at indexing time or as part of query processing (like stopwords)
Stemming

- Generally a small but significant effectiveness improvement
  - can be crucial for some languages
  - e.g., 5-10% improvement for English, up to 50% in Arabic

<table>
<thead>
<tr>
<th>Word</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitab</td>
<td>a book</td>
</tr>
<tr>
<td>kitabi</td>
<td>my book</td>
</tr>
<tr>
<td>alkitab</td>
<td>the book</td>
</tr>
<tr>
<td>kitabuki</td>
<td>your book (f)</td>
</tr>
<tr>
<td>kitabuka</td>
<td>your book (m)</td>
</tr>
<tr>
<td>kitabuhu</td>
<td>his book</td>
</tr>
<tr>
<td>kataba</td>
<td>to write</td>
</tr>
<tr>
<td>maktaba</td>
<td>library, bookstore</td>
</tr>
<tr>
<td>maktab</td>
<td>office</td>
</tr>
</tbody>
</table>

Words with the Arabic root ktb
Stemming

- Two basic types of stemmers
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words

- Algorithmic stemmers
  - *suffix-s*: remove ‘s’ endings assuming plural
    - e.g., cats → cat, lakes → lake, wiis → wii
    - Many *false negatives*: supplies → supplie
    - Some *false positives*: ups → up

- More complex stemmers add more endings
  - -ing, -ed
  - Fewer false negatives, more false positives
Porter Stemmer

- Algorithmic stemmer used in IR experiments since the 70s
- Consists of a series of rules
  - Find the longest possible suffix at each step
  - Some non-intuitive
- Effective in TREC
- Produces *stems* not *words*
- Makes a number of errors and difficult to modify
Porter Stemmer: Example step (1 of 5)

**Step 1a:**

- Replace *sses* by *ss* (e.g., *stresses* → *stress*).
- Delete *s* if the preceding word part contains a vowel not immediately before the *s* (e.g., *gaps* → *gap* but *gas* → *gas*).
- Replace *ied* or *ies* by *i* if preceded by more than one letter, otherwise by *ie* (e.g., *ties* → *tie*, *cries* → *cri*).
- If suffix is *us* or *ss* do nothing (e.g., *stress* → *stress*).

**Step 1b:**

- Replace *eed*, *eedly* by *ee* if it is in the part of the word after the first non-vowel following a vowel (e.g., *agreed* → *agree*, *feed* → *feed*).
- Delete *ed*, *edly*, *ing*, *ingly* if the preceding word part contains a vowel, and then if the word ends in *at*, *bl*, or *iz* add *e* (e.g., *fished* → *fish*, *pirating* → *pirate*), or if the word ends with a double letter that is not *ll*, *ss* or *zz*, remove the last letter (e.g., *falling* → *fall*, *dripping* → *drip*), or if the word is short, add *e* (e.g., *hoping* → *hope*).
- Whew!
Porter Stemmer

■ Some errors of Porter stemmer

<table>
<thead>
<tr>
<th>False positives</th>
<th>False negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization/organ</td>
<td>european/europe</td>
</tr>
<tr>
<td>generalization/generic</td>
<td>cylinder/cylindrical</td>
</tr>
<tr>
<td>numerical/numerous</td>
<td>matrices/matrix</td>
</tr>
<tr>
<td>policy/police</td>
<td>urgency/urgent</td>
</tr>
<tr>
<td>university/universe</td>
<td>create/creation</td>
</tr>
<tr>
<td>addition/additive</td>
<td>analysis/analyses</td>
</tr>
<tr>
<td>negligible/negligent</td>
<td>useful/usefully</td>
</tr>
<tr>
<td>execute/executive</td>
<td>noise/noisy</td>
</tr>
<tr>
<td>past/paste</td>
<td>decompose/decomposition</td>
</tr>
<tr>
<td>ignore/ignorant</td>
<td>sparse/sparsity</td>
</tr>
<tr>
<td>special/specialized</td>
<td>resolve/resolution</td>
</tr>
<tr>
<td>head/heading</td>
<td>triangle/triangular</td>
</tr>
</tbody>
</table>

■ Porter2 stemmer addresses some of these issues

■ Approach has been used with other languages
Dictionary-based Stemmers

- Word-relationships stored explicitly
- Difficult cases are caught
  - Is, be, was
  - Few false positives
- But: Language evolves
- Observation:
  - Old words are irregular
  - Newer words are more regular
- Thus: Hybrid approach
  - Dictionary-based for old words
  - Algorithmic-based for new words
Krovetz Stemmer

- Hybrid algorithmic-dictionary
  - Word checked in dictionary
    - If present, either left alone or replaced with “exception”
    - If not present, word is checked for suffixes that could be removed
    - After removal, dictionary is checked again
    - If still not present, different endings are tried
- Produces words not stems
- Comparable effectiveness
- Lower false positive rate, somewhat higher false negative
Stemmer Comparison

- Original text
  - Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

- Porter stemmer
  - document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

- Krovetz stemmer
  - document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochemic pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale
Many queries are 2-3 word phrases

- Phrases are
  - More precise than single words
    - e.g., documents containing “black sea” vs. two words “black” and “sea”
  - Less ambiguous
    - e.g., “big apple” vs. “rotten apple” vs. “apple”

- Can be difficult for ranking
  - e.g., Given query “fishing supplies”, how do we score documents with
    - exact phrase many times
    - exact phrase just once
    - individual words in same sentence, same paragraph, whole document
    - variations on words?
Phrases

- Ranking: See retrieval model
  - But: Deal with phrases during text processing?
- Text processing issue – how are phrases recognized?
- Three possible approaches:
  - Identify syntactic phrases using a part-of-speech (POS) tagger.
  - Use word $n$-grams.
  - Store word positions in indexes and use $proximity$ operators in queries.
POS Tagging

- POS taggers use statistical models or rule-based models of text to predict syntactic tags of words
- Trained on large corpora
  - Example tags:
    - NN (singular noun), NNS (plural noun), VB (verb), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).
- Phrases can then be defined as simple noun groups (noun phrase)
  - Or simpler: Sequence of nouns, or nouns plus adjective
- Disadvantage: Slow
Original text
- Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger
- Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.
## Example Noun Phrases

<table>
<thead>
<tr>
<th>TREC data</th>
<th>Patent data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>Frequency</td>
</tr>
<tr>
<td>65824</td>
<td>975362</td>
</tr>
<tr>
<td>61327</td>
<td>191625</td>
</tr>
<tr>
<td>33864</td>
<td>147352</td>
</tr>
<tr>
<td>18062</td>
<td>95097</td>
</tr>
<tr>
<td>17788</td>
<td>87903</td>
</tr>
<tr>
<td>17308</td>
<td>81809</td>
</tr>
<tr>
<td>15513</td>
<td>78458</td>
</tr>
<tr>
<td>15009</td>
<td>75850</td>
</tr>
<tr>
<td>12869</td>
<td>66407</td>
</tr>
<tr>
<td>12799</td>
<td>59828</td>
</tr>
<tr>
<td>12067</td>
<td>58724</td>
</tr>
<tr>
<td>10811</td>
<td>56715</td>
</tr>
<tr>
<td>9912</td>
<td>34619</td>
</tr>
<tr>
<td>8127</td>
<td>54117</td>
</tr>
<tr>
<td>7640</td>
<td>52195</td>
</tr>
<tr>
<td>7620</td>
<td>52003</td>
</tr>
<tr>
<td>7524</td>
<td>46299</td>
</tr>
<tr>
<td>7436</td>
<td>41694</td>
</tr>
<tr>
<td>7362</td>
<td>40554</td>
</tr>
<tr>
<td>7086</td>
<td>37911</td>
</tr>
<tr>
<td>6792</td>
<td>35827</td>
</tr>
<tr>
<td>6348</td>
<td>34881</td>
</tr>
<tr>
<td>6157</td>
<td>33947</td>
</tr>
<tr>
<td>5955</td>
<td>32338</td>
</tr>
<tr>
<td>5837</td>
<td>30193</td>
</tr>
</tbody>
</table>
Word positions

- POS tagging too slow for large collections
- Instead: Store word position information in index
- Identify phrases only when query is processed
- More flexible in types of phrases
  - Not restricted to adjacent words
  - Identification of phrases using proximity / occurrence within a window

- Indexing positions and retrieval model for positions: Later
Word N-Grams

- Simpler definition – phrase is any sequence of \( n \) words – known as \( n \)-grams
  - *bigram*: 2 word sequence, *trigram*: 3 word sequence, *unigram*: single words
  - \( n \)-grams also used at character level for applications such as OCR
  - Also useful for indexing Chinese text
- \( n \)-grams typically formed from *overlapping* sequences of words
  - i.e. move \( n \)-word “window” one word at a time in document
- Indexes grow larger
Frequent n-grams are more likely to be meaningful phrases

N-grams form a Zipf distribution
  - Better fit than words alone (if all n-grams in one pot)

Could index all n-grams up to specific length
  - Much faster than POS tagging
  - Uses a lot of storage:
    - Document containing 1,000 words would contain 3,990 instances of word n-grams of length $2 \leq n \leq 5$
  - Remove stopword n-grams: “and the”, “there is”, ...
    - But again: “to be or not to be”
Google N-Grams

“All Our N-gram are Belong to You”

- Web search engines index n-grams
- Google sample (http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html):
  - Number of tokens: 1,024,908,267,229
  - Number of sentences: 95,119,665,584
  - Number of unigrams: 13,588,391
  - Number of bigrams: 314,843,401
  - Number of trigrams: 977,069,902
  - Number of fourgrams: 1,313,818,354
  - Number of fivegrams: 1,176,470,663
- Most frequent trigram in English is “all rights reserved”
  - In Chinese, “limited liability corporation”
  - Not dominated by patterns of speech (“and will be”)
Some parts of documents are more important than others.

- Similar to databases: Column-names

Document parser recognizes structure using markup, such as HTML tags

- Headers, anchor text, bolded text all likely to be important
- Metadata can also be important
- Links used for link analysis

**Tropical fish**

From Wikipedia, the free encyclopedia

*Tropical fish* include fish found in tropical environments around the world, including both freshwater and saltwater species. Fishkeepers often use the term *tropical fish* to refer only those requiring fresh water, with saltwater tropical fish referred to as *marine fish*.

Tropical fish are popular aquarium fish, due to their often bright coloration. In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.
<html>
<head>
<meta name="keywords" content="Tropical fish, Airstone, Albinism, Algae eater, Aquarium, Aquarium fish feeder, Aquarium furniture, Aquascaping, Bath treatment (fishkeeping), Berlin Method, Biotope"/>

<title>Tropical fish - Wikipedia, the free encyclopedia</title>
</head>
<body>

<h1 class="firstHeading">Tropical fish</h1>

<p>Tropical fish include <a href="/wiki/Fish" title="Fish">fish</a> found in <a href="/wiki/Tropics" title="Tropics">tropical</a> environments around the world, including both <a href="/wiki/Fresh_water" title="Fresh water">fresh water</a> and <a href="/wiki/Sea_water" title="Sea water">salt water</a> species. <a href="/wiki/Fishkeeping" title="Fishkeeping">Fishkeepers</a> often use the term <i>tropical fish</i> to refer only those requiring fresh water, with saltwater tropical fish referred to as <i>a</i><a href="/wiki/List_of_marine_aquarium_fish_species" title="List of marine aquarium fish species">marine fish</a>.<p>

<p>Tropical fish are popular <a href="/wiki/Aquarium" title="Aquarium">aquarium</a> fish, due to their often bright coloration. In freshwater fish, this coloration typically derives from <a href="/wiki/Iridescence" title="Iridescence">iridescence</a>, while saltwater fish are generally <a href="/wiki/Pigment" title="Pigment">pigmented</a>.</p>

</body></html>
Document Structure and Markup

- URL itself is source for words
- Depth of URL: Where is IBM’s homepage?
  - www.ibm.com vs.
  - www.pcworld.com/businesscenter/article/698/ibm_buys_apt!

- HTML for layout and presentation
- XML for semantic markup
  - Simple **Dublin Core Metadata Element Set**
    - Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, Rights
  - Geotagging
    - `<meta name="geo.position" content="50.167958;-97.133185">`
    - `<meta name="geo.placename" content="Rockwood Rural Municipality, Manitoba, Canada">`
    - `<meta name="geo.region" content="ca-mb">`
Overview

- Text statistics
- Document parsing
- Link Analysis
- Information Extraction
Link Analysis

- Links are a key component of the Web.
  - Relationships
- Important for navigation, but also for search
  - e.g., `<a href="http://example.com">Example website</a>`
  - “Example website” is the anchor text.
  - “http://example.com” is the destination link.
  - Both are used by search engines.
- No relevance for desktop search
Anchor Text

- Used as a description of the content of the destination page
  - Collection of anchor texts in all links pointing to a page used as an additional text field
- Anchor text tends to be short, descriptive, and similar to query text.
  - `<a href="www.ebay.com">ebay</a>`
  - But: `<a href="www.ebay.com">click here</a>`
- Written by people who are not author of page
  - Description from a different perspective
  - Description of most important aspect
- Link itself is also a vote for importance
- Retrieval experiments have shown that anchor text has significant impact on effectiveness for some types of queries.
  - Especially homepages
  - More effective than PageRank
PageRank

- Tens of billions of web pages, some more informative than others
  - Spam vs. personal homepage/photo album vs. news site vs. corporate homepage
  - Ranking difficult
- Links can be viewed as information about the popularity (authority?) of a web page
  - Can be used by ranking algorithm
- Inlink count could be used as simple measure
  - Susceptible to link spam
- Link analysis algorithms like PageRank provide more reliable ratings
  - Less susceptible to link spam
PageRank: Random Surfer
Surfer Bob is bored

Surprise me!

Random link
PageRank: Random Surfer Model

- Browse the Web using the following algorithm:
  - Choose a random number $r$ between 0 and 1
  - If $r < \lambda$:
    - Go to a random page
  - If $r \geq \lambda$:
    - Click a link at random on the current page
  - Start again

- PageRank of a page is the probability that the “random surfer” will be looking at that page
  - Links from popular pages will increase PageRank of pages they point to, because they are more often visited than non-popular pages
  - Many pages will be reached very often (thousands of time more often than others)

- $\lambda$ is typically small
Dangling Links

- Random jump guarantees that every page will be reached at some point in time.
- Random jump prevents getting stuck on pages that
  - do not have links,
  - contain only links that no longer point to other pages, or
  - have links forming a loop.
- Links that point to the first two types of pages are called **dangling links**.
  - May also be links to pages that have not yet been crawled

- Problem: Bob does not have enough time...
PageRank

- PageRank ($PR$) of page C:
  \[ PR(C) = \frac{PR(A)}{2} + \frac{PR(B)}{1} \]

- More generally,
  \[ PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v} \]

- where $B_u$ is the set of pages that point to $u$, and $L_v$ is the number of outgoing links from page $v$ (not counting duplicate links)
- But: What is $PR(v)$?
Don’t know PageRank values at start
Assume equal values (1/3 in this case), then iterate:

- First iteration:
  \[ PR(C) = \frac{0.33}{2} + 0.33 = 0.5, \quad PR(A) = 0.33, \quad PR(B) = 0.17 \]

- Second iteration:
  \[ PR(C) = \frac{0.33}{2} + 0.17 = 0.33, \quad PR(A) = 0.5, \quad PR(B) = 0.17 \]

- Third iteration:
  \[ PR(C) = 0.42, \quad PR(A) = 0.33, \quad PR(B) = 0.25 \]

Converges to \( PR(C) = 0.4, \quad PR(A) = 0.4, \) and \( PR(B) = 0.2 \)
PageRank

- Taking random page jump into account, 1/3 chance of going to any page when \( r < \lambda \)
- \( PR(C) = \lambda \cdot \frac{1}{3} + (1 - \lambda) \cdot (PR(A)/2 + PR(B)/1) \)
- More generally,

\[
PR(u) = \frac{\lambda}{N} + (1 - \lambda) \cdot \sum_{v \in B_u} \frac{PR(v)}{L_v}
\]

- where \( N \) is the number of pages, \( \lambda \) typically 0.15
- Equivalent to \( R = T \cdot R \)
  - Where \( R \) is vector of PageRank values and \( T \) is transition probability matrix:

\[
T_{ij} = \frac{\lambda}{N} + (1 - \lambda) \cdot \frac{1}{L_i}
\]

- \( R \) is Eigenvector of \( T \)
procedure PAGE\text{RANK}(G)
\begin{itemize}
\item $G$ is the web graph, consisting of vertices (pages) and edges (links).
\end{itemize}
\begin{itemize}
\item $(P, L) \leftarrow G$ \Comment{Split graph into pages and links}
\end{itemize}
\begin{itemize}
\item $I \leftarrow$ a vector of length $|P|$ \Comment{The current PageRank estimate}
\end{itemize}
\begin{itemize}
\item $R \leftarrow$ a vector of length $|P|$ \Comment{The resulting better PageRank estimate}
\end{itemize}
\begin{itemize}
\item for all entries $I_i \in I$ do
\begin{itemize}
\item $I_i \leftarrow 1/|P|$ \Comment{Start with each page being equally likely}
\end{itemize}
\end{itemize}
\begin{itemize}
\item while $R$ has not converged do
\end{itemize}
\begin{itemize}
\item for all entries $R_i \in R$ do
\end{itemize}
\begin{itemize}
\item $R_i \leftarrow \lambda/|P|$ \Comment{Each page has a $\lambda/|P|$ chance of random selection}
\end{itemize}
\begin{itemize}
\item end for
\end{itemize}
\begin{itemize}
\item for all pages $p \in P$ do
\begin{itemize}
\item $Q \leftarrow$ the set of pages $p$ such that $(p, q) \in L$ and $q \in P$
\end{itemize}
\begin{itemize}
\item if $|Q| > 0$ then
\end{itemize}
\begin{itemize}
\item for all pages $q \in Q$ do
\end{itemize}
\begin{itemize}
\item $R_q \leftarrow R_q + (1 - \lambda)I_p/|Q|$ \Comment{Probability $I_p$ of being at page $p$}
\end{itemize}
\begin{itemize}
\item end for
\end{itemize}
\begin{itemize}
\item else
\end{itemize}
\begin{itemize}
\item for all pages $q \in P$ do
\end{itemize}
\begin{itemize}
\item $R_p \leftarrow R_q + (1 - \lambda)I_p/|P|$
\end{itemize}
\begin{itemize}
\item end for
\end{itemize}
\begin{itemize}
\item end if
\end{itemize}
\begin{itemize}
\item $I \leftarrow R$ \Comment{Update our current PageRank estimate}
\end{itemize}
\begin{itemize}
\item end for
\end{itemize}
\begin{itemize}
\item end while
\end{itemize}
\begin{itemize}
\item return $R$
\end{itemize}
\begin{itemize}
\item end procedure
\end{itemize}
Link Quality

- Link quality is affected by spam and other factors
  - e.g., *link farms* to increase PageRank
  - *trackback links* in blogs can create loops
  - Links from comments section of popular blogs boost own web page
    - Blog services modify comment links to contain `rel=nofollow` attribute
    - e.g., “Come visit my `<a rel=nofollow href="http://www.page.com">web page</a>`.”
Trackbacks are a fundamentally different kind of link.
Overview

- Text statistics
- Document parsing
- Link Analysis
- Information Extraction
Information Extraction

- Automatically extract structure from text
  - Annotate document using tags to identify extracted structure
  - Near-term goal: Improve ranking
  - Far-term goal: Turn search problem into database problem

- Already some information extraction
  - HTML structure
  - XML annotations

- Named entity recognition (NER)
  - Identify word or sequence of words that refer to something of interest in a particular application.
  - e.g., people, companies, locations, dates, product names, prices, drug names, etc.
  - Also: Semantic annotation (domain-specific)
“Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of tropical fish.”

Example shows semantic annotation of text using XML tags

Information extraction also includes document structure and more complex features such as relationships and events

Uses

- Faceted search
- Improved browsing (clickable locations, phone-numbers, etc.)
Named Entity Recognition

- **Rule-based**
  - Uses *lexicons* (lists of words and phrases) that categorize names
    - e.g., locations, person names, organizations, etc.
  - Rules (patterns) also used to verify or find new entity names, e.g.,
    - “<number> <word> street” for addresses
    - “<street address>, <city>” or “in <city>” to verify city names
    - “<street address>, <city>, <state>” to find new cities
    - “<title> <name>” to find new names
  - Rules either developed manually by trial and error or using machine learning techniques
Named Entity Recognition

- **Statistical**
  - Uses a probabilistic model of the words in and around an entity
  - Probabilities estimated using *training data* (manually annotated text)
  - Hidden Markov Model (HMM) is one approach

- **HMM for Extraction**
  - Resolve ambiguity (homonyms) in a word using *context*
    - Like humans
    - e.g., “marathon” is a location or a sporting event, “boston marathon” is a specific sporting event
  - Model the context using a *generative* model of the sequence of words
    - *Markov property*: the next word in a sequence depends only on a small number of the previous words
Markov Model describes a process as a collection of states with transitions between them.

- Each transition has a probability associated with it.
- Next state depends only on current state and transition probabilities.

Hidden Markov Model

- Each state has a set of possible outputs.
- Outputs have probabilities.
- “Hidden”, because sequence of states not visible
  - Output is visible, however
Each state is associated with a probability distribution over words (the output)
HMM for Extraction

- Could generate sentences with this model
- To recognize named entities, find sequence of “labels” that give highest probability for the sentence
  - Only the outputs (words) are visible or observed, states are “hidden”.
  - “Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of tropical fish.”
  - <start><name><not-an-entity><location><not-an-entity><end>
- **Viterbi** algorithm used for recognition
  - Dynamic programming
Named Entity Recognition

- Accurate recognition requires about 1 million words of training data (1,500 news stories)
  - May be more expensive than developing rules for some applications
- Both rule-based and statistical can achieve about 90% effectiveness for categories such as names, locations, organizations.
  - Others, such as product name, can be much worse
Internationalization

- 2/3 of the Web is in English
  - But decreasing
- About 50% of Web users do not use English as their primary language
- Many (maybe most) search applications have to deal with multiple languages
  - *monolingual search*: search in one language, but with many possible languages
  - *cross-language search*: search in multiple languages at the same time
Internationalization

- Many aspects of search engines are language-neutral
- Major differences are in text processing:
  - Text encoding (converting to Unicode)
  - Tokenizing (many languages have no word separators)
  - Stemming
- Cultural differences may also impact interface design and features provided
Auch im Deutschen

- Donaudampfschifffahrts-gesellschaft

1. Original text
旱灾在中国造成的影响
(the impact of droughts in China)

2. Word segmentation
旱灾 在 中国 造成 的 影响
drought at china make impact

3. Bigrams
旱灾 灾在 在中 中国 国造
造成 成的 的影 影响

Donaudampfschifffahrtselectrizitätengemäßbetriebswerkbaufachbeamtengesellschaft

Donaudampf


Während Donaudampfschifffahrtsgesellschaft (auch „Erste Donau-Dampfschifffahrts-Gesellschaft“, abgekürzt DDSG oder EDDG) als Name einer 1829 gegründeten Gesellschaft, die von 1830 bis 1996 Personen- und Güterfähre on der Donau betrieb, historisch belegt ist, sind für die besagte „Unterbeamtengesellschaft“ keine Belege dafür bekannt, dass jemals eine Gesellschaft dieses Namens existierte und es sich bei diesem Namen nicht bloß um ein Kunstwort handelt, das zur Erzielung einer besonderen Wortlänge erzeugt wurde.

Amtlich belegt ist hingegen das Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz in Mecklenburg-Vorpommern.