Chapter 4

Processing Text
Processing Text

- Modifying/Converting *documents* to *index terms*
  - Convert the many forms of *words* into more consistent *index terms* that represent the content of a document

- What are the problems?
  - Matching the *exact* string of characters typed by the user is too restrictive, e.g., case-sensitivity, punctuation, stemming
    - it doesn’t work very well in terms of *effectiveness*
  - Sometimes not clear where words begin and end
    - Not even clear what a word is in some languages, e.g., in Chinese and Korean
  - *Not* all words are of *equal value* in a search, and understanding the *statistical* nature of text is critical
Indexing Process

- Identifies and stores documents for indexing
- Text + Meta data (Doc type, structure, features, size, etc.)
- Text Acquisition
- Index Creation
- Document data store
- Takes index terms and creates data structures (indexes) to support fast searching
- Text Transformation or Text Processing
- Transforms documents into index terms or features
- E-mail, Web pages, News articles, Memos, Letters
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/significant words* occur often in documents but are *not high frequency* in collection
Zipf’s Law

- Distribution of word frequencies is very skewed
  - Few words occur very often, many hardly ever occur
  - e.g., “the” and “of”, two common words, make up about 10% of all word occurrences in text documents

- Zipf’s law:
  - The frequency $f$ of a word in a corpus is inversely proportional to its rank $r$ (assuming words are ranked in order of decreasing frequency)

$$f = \frac{k}{r} \equiv f \times r = k$$

where $k$ is a constant for the corpus
# 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>0.079</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>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>
Example. Zipf’s law for AP89 with problems at high and low frequencies

According to [Ha 02], Zipf’s law

- does not hold for rank > 5,000
- is valid when considering single words as well as $n$-gram phrases, combined in a single curve.

Vocabulary Growth

- **Heaps’ Law**, another prediction of *word occurrence*
- As *corpus* grows, so does *vocabulary size*. However, *fewer* new words when corpus is already *large*
- Observed relationship (**Heaps’ Law**):

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

  where

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

  - Predicting that the number of *new* words increases very rapidly when the corpus is *small*
AP89 Example (40 million words)

\[ v = k \times n^\beta \]
Heaps’ Law Predictions

- Number of **new** words *increases* very rapidly when the corpus is **small**, and continue to increase indefinitely.

- 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., < 1000) are much worse.
Heaps’ Law on the Web

- Heaps’ Law works with very large corpora
  - New words occurring even after seeing 30 million!
  - Parameter values 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
Heaps’ Law vs. Zipf’s Law

- As stated in [French 02]:
  - The observed vocabulary growth has a positive correlation with Heaps’ law
  - Zipf’s law, on the other hand, is a poor predictor of high-ranked terms, i.e., Zipf’s law is adequate for predicting medium to low ranked terms
  - While Heaps’ law is a valid model for vocabulary growth of web data, Zipf’s law is not strongly correlated with web data

Estimating Result Set Size

- **Word occurrence statistics** can be used to estimate the size of the results from a web search.
- How many pages (in the results) contain all of the query terms (based on word occurrence statistics)?
- **Example.** For the query “a b c”:

\[ f_{abc} = N \times \frac{f_a}{N} \times \frac{f_b}{N} \times \frac{f_c}{N} = \frac{f_a \times f_b \times f_c}{N^2} \]

- \( f_{abc} \): estimated size of the result set using joint probability
- \( f_a, f_b, f_c \): the number of documents that terms a, b, and c occur in, respectively
- \( N \) is the total number of documents in the collection
- Assuming that terms occur independently
Collection size ($N$) is 25,205,179
Estimating Collection Size

- Important issue for Web search engines, in terms of coverage

- Simple method: use *independence* model, even not realizable
  
  - Given two words, \(a\) and \(b\), that are independent, and \(N\) is the *estimated size* of the document collection
    
    \[
    \frac{f_{ab}}{N} = \frac{f_a}{N} \times \frac{f_b}{N}
    \]
    
    \[\iff \]
    
    \[N = \frac{(f_a \times f_b)}{f_{ab}}\]

- Example. For GOV2
  
  \[
  f_{lincoln} = 771,326
  
  f_{tropical} = 120,990
  
  f_{lincoln \cap tropical} = 3,018
  
  N = \frac{(120,990 \times 771,326)}{3,018}
  
  = 30,922,045 (actual number is 25,205,179)\]
Result Set Size Estimation

- Poor estimates because words are not independent
- Better estimates possible if co-occurrence info. available

\[ P(a \cap b \cap c) = P(a \cap b) \times P(c \mid a \cap b) \]

\[ = P(a \cap b) \times (P(b \cap c) / P(b)) \]

\( f_{\text{tropical}} \cap \text{aquarium} \cap \text{fish} = f_{\text{tropical}} \cap \text{aquarium} \times f_{\text{aquarium}} \cap \text{fish} / f_{\text{aquarium}} \]

\[ = 1921 \times 9722 / 26480 \]

\[ = 705 (1,529, \text{actual}) \]

\( f_{\text{tropical}} \cap \text{breeding} \cap \text{fish} = f_{\text{tropical}} \cap \text{breeding} \times f_{\text{breeding}} \cap \text{fish} / f_{\text{breeding}} \]

\[ = 5510 \times 36427 / 81885 \]

\[ = 2,451 (3,629 \text{ actual}) \]
Result Set Estimation

Even **better estimates** using *initial result set* (word frequency + current result set)

- Estimate is simply \( C/s \)
  - where \( s \) is the proportion of the total number of *documents* that have been *ranked* (i.e., processed) & \( C \) is the number of documents found that contain all the *query words*

- **Example.** “*tropical fish aquarium*” in GOV2
  - After processing 3,000 out of the 26,480 documents that contain “aquarium”, \( C = 258 \)
    \[
    f_{tropical \cap fish \cap aquarium} = \frac{258}{(3000 \div 26480)} = 2.277 \ (> 1.529) \]
  - After processing 20% of the documents, \( f_{tropical \cap fish \cap aquarium} = 1.778 \) (1,529 is real value)
    \[
    f_{tropical \cap fish \cap aquarium} = 1.778 (1,529 \text{ is real value})
    \]
    where \( C = 356 \) & 5,296 documents have been ranked
Tokenizing

- Forming *words* from *sequence of characters*
- Surprisingly complex in English, can be harder in other languages
- Tokenization strategy of early IR systems:
  - Any sequence of alphanumeric characters of length 3 or more
  - Terminated by a space or other special character
  - Upper-case changed to lower-case
Tokenizing

- **Example (Using the Early IR Approach).**
  - “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 the *effectiveness* of some queries
Tokenizing Problems

- Small words can be important in some queries, usually in combinations
  - *xp, bi, pm, cm, el paso, kg, ben e king, master p, 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, House, Senior, Time, Key

- **Apostrophes** can be a part of a word/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
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

- First step is to use parser to identify appropriate parts of doc (e.g., markup/tags in markup language) to tokenize.

- An approach: defer complex decisions to other components, such as stopping, stemming, & query transformation.
  - Word is any sequence of alphanumeric characters, terminated by a space or special character, converted to lower-case.
  - Everything indexed.
  - Example: 92.3 → 92 3 but search finds document with 92 and 3 adjacent.
  - To enhance the effectiveness of query transformation, incorporate some rules into the tokenizer to reduce dependence on other transformation components.
Tokenizing Process

- Not that different than simple tokenizing process used in the 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. → phd
Stopping

- Function words (conjunctions, prepositions, articles) have little meaning on their own.

- High occurrence frequencies.

- Treated as stopwords (i.e., text processing stops when words are detected & removed hereafter).
  - Reduce index space
  - Improve response time
  - Improve effectiveness

- Can be important in combinations.
  - e.g., “to be or not to be”
Stopword list can be created from high-frequency words or based on a standard list.

Lists are customized for applications, domains, and even parts of documents.

- e.g., “click” is a good stopword for anchor text

One of the policies is to index all words in docs, & make decisions about which words to use at query time.
Stemming

- Many *morphological variations* of words to convey a single idea
  - *Inflectional* (plurals, tenses)
  - *Derivational* (making verbs into nouns, etc.)
- 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/as part of query processing (like stopwords)
Stemming

Two basic types

- **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
  - Some *false positives*: ups → up (find a relationship when none exists)
  - Many *false negatives*: countries → countrie (Fail to find term relationship)
Porter Stemmer

- Algorithmic stemmer used in IR experiments since the 70’s
- Consists of a series of rules designed to extract the *longest possible suffix* at each step, e.g.,

  **Step 1a:**
  - Replace *sses* by *ss* (e.g., *stresses* → *stress*)
  - Delete *s* if the preceding word contains a *vowel* not immediately before *s* (e.g., *gaps* → *gap*, *gas* → *gas*)
  - Replace *ied* or *ies* by *i* if preceded by > 1 letter; o.w., by *ie* (e.g., *ties* → *tie*, *cries* → *cri*)

- Effective in TREC
- Produces *stems* not *words*
- Makes a number of *errors* and *difficult* to *modify*
Errors of Porter Stemmer

- It is difficult to capture all the subtleties of a language in a simple algorithm

- Porter2 stemmer addresses some of these issues

- Approach has been used with other languages

<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>
Link Analysis

- **Links** are a key component of the Web
- 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
Anchor Text

- Describe the content of the destination page
  - i.e., collection of anchor text 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

- Retrieval experiments have shown that anchor text has significant impact on effectiveness for some types of queries
  - i.e., more than PageRank
PageRank

- Billions of web pages, some more informative than others
- Links can be viewed as *information* about the *popularity* (authority?) of a web page
  - Can be used by *ranking algorithms*
- *Inlink count* could be used as simple measure
- Link analysis algorithms like PageRank provide more reliable ratings, but *Less susceptible* to link spam
- PageRank of a page is the *probability* that the “random surfer” will be looking at that page
  - Links from *popular* pages increase PageRank of pages they point to, i.e., links tend to point to popular pages
PageRank

- **PageRank (PR) of page** $C = \text{PR}(A)/2 + \text{PR}(B)/1$

- More generally

\[
\text{PR}(u) = \sum_{v \in B_u} \frac{\text{PR}(v)}{L_v}
\]

- where $u$ is a web page
- $B_u$ is the set of pages that point to $u$
- $L_v$ is the number of outgoing links from page $v$
  (not counting duplicate links)
PageRank

- Don’t know *PageRank values* at start

- **Example.** Assume equal values of 1/3, then
  
  - 1st iteration: \( PR(C) = \frac{0.33}{2} + \frac{0.33}{1} = 0.5 \)
    - \( PR(A) = \frac{0.33}{1} = 0.33 \)
    - \( PR(B) = \frac{0.33}{2} = 0.17 \)
  
  - 2nd iteration: \( PR(C) = \frac{0.33}{2} + \frac{0.17}{1} = 0.33 \)
    - \( PR(A) = \frac{0.5}{1} = 0.5 \)
    - \( PR(B) = \frac{0.33}{2} = 0.17 \)
  
  - 3rd iteration: \( PR(C) = \frac{0.5}{2} + \frac{0.17}{1} = 0.42 \)
    - \( PR(A) = \frac{0.33}{1} = 0.33 \)
    - \( PR(B) = \frac{0.5}{2} = 0.25 \)

- Converges to \( PR(C) = 0.4 \)
  
  - \( PR(A) = 0.4 \)
  
  - \( PR(B) = 0.2 \)