Indexing

Info 427

Announcements

• Assignment 2 released
  – Due on Monday Oct 21, 11:59PM
  – Lab 6 and 7 partial solutions available on Wiki

• A1 grades released later today

Indexing

Zipf’s Law

• Distribution of word frequencies is very skewed
  – a few words occur often, most words rarely occur

• Zipf’s “law”
  – observation that the frequency of a word in roughly proportional to 1/r, where r is its rank
  – E.g. the most common word, “the”, occurs...
    • twice as often as the second most common word ("of")
    • 3x as often as the third most common word ("and")
    • 10x as often as the tenth most common word ("it")

Zipf’s Law

Power law distribution \( f(x) = ax^k \)


Zipf’s Law

Power law distribution

Plotting on regular (linear) axes

Plotting on log-log axes:

Contrast with the Uniform Distribution: \( f(x) = a \)
Power law distribution $f(x) = ax^k$

- Plotted on regular (linear) axes:
- Plotted on log-log axes:

Contrast with the Gaussian (Normal) Distribution:

- Power law distribution
- Example:
  - Moby Dick scientific papers 1981-1997
  - AOL users visiting sites '97
  - books sold bestsellers 1895-1965
  - telephone calls received AT&T customers on 1 day
  - earthquake magnitude California 1900-1992

Zipf's Law for AP89

- Note imperfect fit at high and low frequencies

Stopping

- Some words occur very frequently, but have little meaning on their own
  - e.g., “20% of all words are 1 of: the, of, to, a, and, in, be”

- Search engines treat these as stopwords
  - ignore during indexing
  - reduce index space, improve response time, improve effectiveness

- Can be important in combinations
  - e.g., “to be or not to be”
Choosing stop words

- 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
- In practice, search engines index all words, choosing which words to treat as stop words at query time

Estimating Result Set Size

• How many pages contain all of the query terms?
  - For the query “a b c”, approximate as:
    \[ f_{abc} = \frac{N}{f_a \cdot f_b \cdot f_c} \]
    where \( f_a, f_b, f_c \) are the number of documents containing terms \( a, b, \) and \( c \), and \( N \) is the number of documents in the collection

GOV2 Example

<table>
<thead>
<tr>
<th>Word(s)</th>
<th>Document Frequency</th>
<th>Estimated Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>tropical</td>
<td>120,900</td>
<td></td>
</tr>
<tr>
<td>fish</td>
<td>1,231,265</td>
<td></td>
</tr>
<tr>
<td>breeding</td>
<td>62,654</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>201</td>
</tr>
<tr>
<td>fish aquarium</td>
<td>9,722</td>
<td>1,188</td>
</tr>
<tr>
<td>fish bredding</td>
<td>36,425</td>
<td>1,077</td>
</tr>
<tr>
<td>aquarium breeding</td>
<td>1,448</td>
<td>86</td>
</tr>
<tr>
<td>tropical fish aquarium</td>
<td>1,239</td>
<td>6</td>
</tr>
<tr>
<td>tropical fish bredding</td>
<td>3,029</td>
<td>15</td>
</tr>
</tbody>
</table>

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

Result Set Size Estimation

• Poor estimates because words are not independent
  - Better estimates possible if co-occurrence information available; i.e., compute statistics for pairs of words
    \[ f_{tropical\cap fish\cap aquarium} \approx f_{tropical\cap aquarium} \cdot f_{fish} = 705 \]
  - Could get even better estimates if we compute statistics for triples, quadruples, etc. of words; why not do this?

Tokenizing

• Tokenization: Breaking a document into index terms (words)
• Surprisingly complex in English, even harder in other languages
• Early Information Retrieval systems used:
  - any sequence of alphanumeric characters of length 3 or more
  - terminated by a space or other special character
  - upper-case changed to lower-case
Tokenization Problems

- **Small words** can be important in some queries, usually in combinations
  - xp, ma, pm, ben e king, el paso, gm, j lo, world war II
- Both hyphenated and non-hyphenated forms of many words are common
  - Sometimes hyphen is not needed
  - 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

- **Special characters** are an important part of tags, URLs, code in documents
- **Capital letters** can change the meanings of words
  - Bush vs bush, Apple vs 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

- **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., b.s., cs.indiana.edu, f.e.a.r.

Tokenizing Process

- Use tokenizer on both documents and queries
  - To break them into index terms – e.g. words
- **Typical tokenization rules**
  - word is any sequence of alphanumeric characters, terminated by a space or special character, with everything converted to lower-case
  - Remove most punctuation (e.g. apostrophes, hyphens, decimal points)
  - Use some rules to handle special cases

Indexing words

- Let’s say someone enters a query like:
  - We’d like to find documents not only with this set of words, but also of very closely related words, e.g.
    - Bush cuts taxes
    - Bush's tax cuts
    - Bush cut taxes
    - Bush cut taxation
    - Bush cutting taxes

Stemming

- Most languages have morphological variations of words that have very similar meanings
  - E.g. plurals, different tenses (inflectional differences)
  - computer vs. computers, test vs. tested, sat vs. sitting
  - E.g. different parts of speech (derivational differences)
  - happy vs. happiness, employ vs. employer
- A stemmer tries to convert words to a common stem
Stemming

- Two basic types
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words
- Algorithmic stemmers
  - Simple example: remove 's' from end of words
    - e.g., cats → cat, lakes → lake, wiis → wii
  - Makes many errors:
    - congress → congres, supplies → suppilie, ups → up, his → hi, etc.

Porter Stemmer

- A commonly used algorithmic stemmer
- Example step (1 of 5)

  Step 1:
  - Replace end by an (e.g., stenm → stem).
  - Delete a if the previous word part contains a vowel and immediately
    before the a (e.g., game → game lost gas → gas).
  - Replace end or one by e if preceded by more than one letter, otherwise
    by le (e.g., firm → firm, item → item).
  - If suff is no or no doing (e.g., stem → stem).

Step 2:
  - Replace end, orily by as if it is in the part of the word after the
    first non-vowel following a vowel (e.g., agreed → agree deal → deal).
  - Delete ed, ildy by mp if the preceding word part contains a
    vowel, and then if the word ends in el, Al, or it ends in, e.g., (ed)ated →
    tated (printing → printed), or if the word ends with a double letter that
    is not, or or, remove the last letter (e.g., lifting → lift, dripping → drip).

Porter Stemmer – some errors

- False positives: Words with different meanings have the same stem
  - university → univers, universe → univers
  - paste → past, past → past
  - Numerical → numer, numerous → numer
- False negatives: Words with similar meanings that have different stems
  - Analysis → analy, analyses → analy
  - Cactus → cactu, cacti → cacti
  - Noise → noisy, noisy → noisy
- Porter2 and other more sophisticated stemmers address some of these issues

Stemming

- Leads to significant improvement in effectiveness
  - e.g., 5-10% improvement for English
  - Crucial for some languages; e.g., 50% improvement for Arabic

<table>
<thead>
<tr>
<th>English</th>
<th>Arabic</th>
</tr>
</thead>
<tbody>
<tr>
<td>kitchen</td>
<td>kitb</td>
</tr>
<tr>
<td>kitbli</td>
<td>ny book</td>
</tr>
<tr>
<td>skilbli</td>
<td>the book</td>
</tr>
<tr>
<td>kitbli (f)</td>
<td>poor book</td>
</tr>
<tr>
<td>kitbli (m)</td>
<td>poor book</td>
</tr>
<tr>
<td>kitbli</td>
<td>free book</td>
</tr>
<tr>
<td>kitbli (f)</td>
<td>to write</td>
</tr>
<tr>
<td>kitbli</td>
<td>library, bookstore</td>
</tr>
<tr>
<td>kitbli (f)</td>
<td>office</td>
</tr>
</tbody>
</table>

Words with the Arabic root kitb

Document Structure and Markup

- Some parts of a documents are more important than others
- Document parser recognizes structure using markup, such as HTML tags
  - Headers, anchor text, bolded text probably important
  - Metadata can also be important
  - Links used for link analysis
Example Web Page

```html
<!doctype html>
<meta name="keywords" content="Tropical fish, Aquarium, Algae, Saltwater"
AquaCulture, Aquarium Fish, Tank, Aquascaping, Fish treatment
(Dechlorination) Shells, Methods, Breeding">
<title>Tropical fish - Wikipedia, the free encyclopedia</title>
<body>
<div class="shell-bordering" title="Tropical fish - Wikipedia">
<p>Tropical fish include fish found in tropical environments
wherever the world, both marine and freshwater species.
Many tropical fish species are kept in home aquariums.
</p>
</div>
</body>
</html>
```

Using HTML tags for weighting

- Some tags might indicate importance of terms in a page:
  - `<TITLE>`
  - Headers: `<H1>`, `<H2>`, …
  - Style: `<strong>`, `<i>`, …
  - Metadata: `<meta name="keywords" ...>`
  - Text color

- Search engines try to detect and ignore text that's gratuitously added to improve search placement
  - Metadata is typically disregarded
  - Invisible text is also typically disregarded

Link Analysis

- Links are a key component of the Web
- Important for navigation, but also for search

```html
<a href="http://example.com">Example website</a>
```

Anchor Text

- Used as a description of 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
Google bombing (GOO.gul bawm.ing) n. Setting up a large number of Web pages with links that point to a specific Web site so that the site will appear near the top of a Google search when users enter the anchor text. —Google bomb n.

Answering queries

• We’ve seen how to parse HTML documents to
  – Extract titles, links, etc.
  – Convert text to stemmed, stopped tokens (words)

• How do we use these information for answering queries?
  – Simple technique: Given a query, scan the entire collection of documents looking for matching pages
Indexes

- **Indexes** are data structures designed to make search faster

- Most common data structure is inverted index
  - "inverted" because it maps words to documents, rather than documents to words
  - similar to a concordance

Inverted Index

- Index terms are usually kept in alphabetical order

- Each index term is associated with an inverted list
  - Contains lists of documents, or lists of word occurrences in documents, and other information
  - Each entry is called a posting
  - The part of the posting that refers to a specific document or location is called a pointer
  - Lists are usually document-ordered (sorted by document number)

Example “Collection”

- $S_1$ Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.
- $S_2$ Fishermen often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.
- $S_3$ Tropical fish are popular aquarium fish, due to their often bright coloration.
- $S_4$ In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Four sentences from the Wikipedia entry for tropical fish

Simple Inverted Index

- Answering queries just involves lookups to the index
  - Which pages contain the word "salt"?
  - Which pages contain "salt" and "water"?
  - Which pages contain "salt" and "water" but not "tropical"?

<table>
<thead>
<tr>
<th>Index Term</th>
<th>Posting Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>salt</td>
<td>Page 1, Page 2, Page 3, Page 4</td>
</tr>
<tr>
<td>water</td>
<td>Page 5, Page 6, Page 7</td>
</tr>
<tr>
<td>tropical</td>
<td>Page 8, Page 9, Page 10</td>
</tr>
<tr>
<td>fresh</td>
<td>Page 11, Page 12, Page 13</td>
</tr>
<tr>
<td>marine</td>
<td>Page 14, Page 15, Page 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Index Term</th>
<th>Posting Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>salt</td>
<td>Page 1, Page 2, Page 3, Page 4</td>
</tr>
<tr>
<td>water</td>
<td>Page 5, Page 6, Page 7</td>
</tr>
<tr>
<td>tropical</td>
<td>Page 8, Page 9, Page 10</td>
</tr>
<tr>
<td>fresh</td>
<td>Page 11, Page 12, Page 13</td>
</tr>
<tr>
<td>marine</td>
<td>Page 14, Page 15, Page 16</td>
</tr>
</tbody>
</table>
Inverted index with counts

- Records # of times word appears in each document
- Supports better ranking algorithms

Example: possible index structure