Text Properties and Languages

Statistical Properties of Text

• How is the frequency of different words distributed?
• How fast does vocabulary size grow with the size of a corpus?
• Such factors affect the performance of information retrieval and can be used to select appropriate term weights and other aspects of an IR system.

Word Frequency

• A few words are very common.
  – 2 most frequent words (e.g. “the”, “of”) can account for about 10% of word occurrences.
• Most words are very rare.
  – Half the words in a corpus appear only once, called *hapax legomena* (Greek for “read only once”)
• Called a “heavy tailed” or “long tailed” distribution, since most of the probability mass is in the “tail” compared to an exponential distribution.
Sample Word Frequency Data
(from B. Croft, UMass)

<table>
<thead>
<tr>
<th>Frequent Word</th>
<th>Number of Occurrences</th>
<th>Percentage of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>7,298,934</td>
<td>5.0</td>
</tr>
<tr>
<td>of</td>
<td>3,893,790</td>
<td>2.7</td>
</tr>
<tr>
<td>to</td>
<td>3,264,653</td>
<td>2.6</td>
</tr>
<tr>
<td>and</td>
<td>2,311,765</td>
<td>1.8</td>
</tr>
<tr>
<td>in</td>
<td>1,598,147</td>
<td>1.2</td>
</tr>
<tr>
<td>for</td>
<td>1,313,561</td>
<td>1.0</td>
</tr>
<tr>
<td>the</td>
<td>1,144,460</td>
<td>0.9</td>
</tr>
<tr>
<td>has</td>
<td>1,066,503</td>
<td>0.8</td>
</tr>
<tr>
<td>said</td>
<td>1,027,113</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Frequencies from 336,310 documents in the 1GB TRIEC Volume 3 Corpus, 125,720,891 total word occurrences, 508,209 unique words

Zipf's Law

- Rank \( (r) \): The numerical position of a word in a list sorted by decreasing frequency \( (f) \).
- Zipf (1949) “discovered” that:
  \[
  f \propto \frac{1}{r} \\
  f \cdot r = k \quad \text{(for constant } k) 
  \]
- If probability of word of rank \( r \) is \( p_r \) and \( N \) is the total number of word occurrences:
  \[
  p_r = \frac{f}{N} = \frac{A}{r} \\
  \text{for corpus indp., const. } A = 0.1 
  \]

Zipf and Term Weighting

- Luhn (1958) suggested that both extremely common and extremely uncommon words were not very useful for indexing.
Prevalence of Zipfian Laws

- Many items exhibit a Zipfian distribution.
  - Population of cities
  - Wealth of individuals
    - Discovered by sociologist/economist Pareto in 1909
  - Popularity of books, movies, music, web-pages, etc.
  - Popularity of consumer products
    - Chris Anderson’s “long tail”

Predicting Occurrence Frequencies

- By Zipf, a word appearing $n$ times has rank $r_n = AN/n$
- Several words may occur $n$ times, assume rank $r_n$ applies to the last of these.
- Therefore, $r_n$ words occur $n$ or more times and $r_{n+1}$ words occur $n+1$ or more times.
- So, the number of words appearing exactly $n$ times is:
  
  $I_n = r_n - r_{n+1} = \frac{AN}{n} - \frac{AN}{n+1} - \frac{AN}{n(n+1)}$

Predicting Word Frequencies (cont)

- Assume highest ranking term occurs once and therefore has rank $D = AN/1$
- Fraction of words with frequency $n$ is:
  
  $I_n = \frac{1}{n(n+1)}$
- Fraction of words appearing only once is therefore $\frac{1}{2}$. 
Occurrence Frequency Data
(from B. Croft, UMass)

<table>
<thead>
<tr>
<th>Number of Occurrences (n)</th>
<th>Predicted Proportion of Occurrences 1/n/(n+1)</th>
<th>Actual Proportion occurring n times L/n</th>
<th>Actual Number of Words occurring n times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.500</td>
<td>.402</td>
<td>204,387</td>
</tr>
<tr>
<td>2</td>
<td>.167</td>
<td>.132</td>
<td>67,082</td>
</tr>
<tr>
<td>3</td>
<td>.081</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>.028</td>
<td>16,332</td>
</tr>
<tr>
<td>6</td>
<td>.024</td>
<td>.021</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>

Frequencies from 136,311 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,269 unique words

Does Real Data Fit Zipf’s Law?

• A law of the form $y = kx^c$ is called a power law.
• Zipf’s law is a power law with $c = -1$
• On a log-log plot, power laws give a straight line with slope $c$.
  \[ \log(y) = \log(kx^c) = \log(k) + c \log(x) \]
• Zipf is quite accurate except for very high and low rank.

Fit to Zipf for Brown Corpus
Mandelbrot (1954) Correction

- The following more general form gives a bit better fit:
  \[ f = p(r + \rho)^{-\beta} \]
  For constants \( P, B, \rho \)

Mandelbrot Fit

Mandelbrot’s function on Brown corpus
\( P = 10^{4-3}, B = 1.15, \rho = 100 \)

Explanations for Zipf’s Law

- Zipf’s explanation was his “principle of least effort.” Balance between speaker’s desire for a small vocabulary and hearer’s desire for a large one.
- Debate (1955-61) between Mandelbrot and H. Simon over explanation.
- Simon explanation is “rich get richer.”
- Li (1992) shows that just random typing of letters including a space will generate “words” with a Zipfian distribution.
  - [http://linkage.rockefeller.edu/wli/zipf/](http://linkage.rockefeller.edu/wli/zipf/)
Zipf’s Law Impact on IR

• **Good News:**
  – Stopwords will account for a large fraction of text so eliminating them greatly reduces inverted-index storage costs.
  – Postings list for most remaining words in the inverted index will be short since they are rare, making retrieval fast.

• **Bad News:**
  – For most words, gathering sufficient data for meaningful statistical analysis (e.g. for correlation analysis for query expansion) is difficult since they are extremely rare.

Vocabulary Growth

• How does the size of the overall vocabulary (number of unique words) grow with the size of the corpus?
• This determines how the size of the inverted index will scale with the size of the corpus.
• Vocabulary not really upper-bounded due to proper names, typos, etc.

Heaps’ Law

• If $V$ is the size of the vocabulary and the $n$ is the length of the corpus in words:
  \[ V = Kn^\beta \]
  with constants $K$, $0 < \beta < 1$

• Typical constants:
  – $K \approx 10–100$
  – $\beta \approx 0.4–0.6$ (approx. square-root)
Heaps’ Law Data

Explanation for Heaps’ Law

• Can be derived from Zipf’s law by assuming documents are generated by randomly sampling words from a Zipfian distribution.

Metadata

• Information about a document that may not be a part of the document itself (data about data).
• Descriptive metadata is external to the meaning of the document:
  – Author
  – Title
  – Source (book, magazine, newspaper, journal)
  – Date
  – ISBN
  – Publisher
  – Length
Metadata (cont)

- **Semantic** metadata concerns the content:
  - Abstract
  - Keywords
  - Subject Codes
    - Library of Congress
    - Dewey Decimal
    - UMLS (Unified Medical Language System)
- Subject terms may come from specific **ontologies** (hierarchical taxonomies of standardized semantic terms).

Web Metadata

- META tag in HTML
  - `<META NAME="keywords" CONTENT="pets, cats, dogs">`
- META “HTTP-EQUIV” attribute allows server or browser to access information:
  - `<META HTTP-EQUIV="expires" CONTENT="Tue, 01 Jan 02">`
  - `<META HTTP-EQUIV="creation-date" CONTENT="23-Sep-01">`

Markup Languages

- Language used to annotate documents with “tags” that indicate layout or semantic information.
- Most document languages (Word, RTF, Latex, HTML) primarily define *layout*.
- History of Generalized Markup Languages:
  - **Standard**
    - GML (1969)
    - SGML (1985)
  - **eXtensible**
    - HTML (1993)
    - XML (1998)
Basic SGML Document Syntax

- Blocks of text surrounded by start and end tags.
  - `<tagname attribute=value attribute=value …>`
  - `</tagname>`
- Tagged blocks can be nested.
- In HTML end tag is not always necessary, but in XML it is.

HTML

- Developed for hypertext on the web.
  - `<a href="http://www.cs.utexas.edu">`
- May include code such as Javascript in Dynamic HTML (DHTML).
- Separates layout somewhat by using style sheets (Cascade Style Sheets, CSS).
- However, primarily defines layout and formatting.

XML

- Like SGML, a metalanguage for defining specific document languages.
- Simplification of original SGML for the web promoted by WWW Consortium (W3C).
- Fully separates semantic information and layout.
- Provides structured data (such as a relational DB) in a document format.
- Replacement for an explicit database schema.
XML (cont)

- Allows programs to easily interpret information in a document, as opposed to HTML intended as layout language for formatting docs for human consumption.
- New tags are defined as needed.
- Structures can be nested arbitrarily deep.
- Separate (optional) Document Type Definition (DTD) defines tags and document grammar.

XML Example

```xml
<person>
  <name>
    <firstname>John</firstname>
    <middlename/>
    <lastname>Doe</lastname>
  </name>
  <age>38</age>
  <email>jdoe@austin.rr.com</email>
</person>

<tag/> is shorthand for empty tag <tag></tag>
Tag names are case-sensitive (unlike HTML)
A tagged piece of text is called an element.
```

XML Example with Attributes

```xml
<product type="food">
  <name language="Spanish">arroz con pollo</name>
  <price currency="peso">2.30</price>
</product>
```

Attribute values must be strings enclosed in quotes.
For a given tag, an attribute name can only appear once.
XML Graph Structures

- Tag “id” and “idref” attributes allows specifying graph-structured data as well as tree-structured data.

```xml
<state id="s2">
    <abbrev>TX</abbrev>
    <name>Texas</name>
</state>
<city id="c2">
    <aircode> AUS </aircode>
    <name> Austin </name>
    <state idref="s2"/>
</city>
```

Document Type Definition (DTD)

- Grammar or schema for defining the tags and structure of a particular document type.
- Allows defining structure of a document element using a regular expression.
- Expression defining an element can be recursive, allowing the expressive power of a context-free grammar.

```xml
<!DOCTYPE db [ 
  <!ELEMENT db (person*)>  
  <!ELEMENT person (name,age,(parent | guardian))>  
  <!ELEMENT name (#PCDATA)>  
  <!ELEMENT age (#PCDATA)>  
  <!ELEMENT parent (person)>  
  <!ELEMENT guardian (person)>  
]>  
```

- *: 0 or more repetitions
- ?: 0 or 1 (optional)
- |: alternation (or)
- PCDATA: Parsed Character Data (may contain tags)
Sample Valid Document for DTD

```xml
<db>
  <person>
    <name>
      <firstname>John</firstname>
      <lastname>Doe</lastname>
    </name>
    <age>26</age>
    <parent>
      <person>
        <name>
          <firstname>Robert</firstname>
          <lastname>Doe</lastname>
        </name>
        <age>55</age>
      </person>
    </parent>
  </person>
</db>
```

DTD (cont)

- Tag attributes are also defined:

  ```xml
  <!ATTLIS name language CDATA #REQUIRED>
  <!ATTLIS price currency CDATA #IMPLIED>
  
  CDATA: Character data (string)
  IMPLIED: Optional
  ```

XSL (Extensible Style-sheet Language)

- Defines layout for XML documents.
- Defines how to translate XML into HTML.
- Define style sheet in document:
  ```xml
  <?xml-stylesheet href="mystyle.css" type="text/css">
  ```
Parsing XML

- Process XML file into an internal data format for further processing.
- SAX (Simple API for XML): Reads the flow of XML text, detecting events (e.g. tag start and end) that are sent back to the application for processing.

DOM

- XML document represented as a tree of Node objects (e.g. Java objects).
- Node class has subclasses:
  - Element
  - Attribute
  - CharacterData
- Node has methods:
  - getParentNode()
  - getChildNodes()

Sample DOM Tree
More Node Methods

- Element node
  - getTagName()
  - getAttributes()
  - getAttribute(String name)
- CharacterData node
  - getData()
- Also methods for adding and deleting nodes and text in the DOM tree, setting attributes, etc.