An Empirical Exploration of Regularities in Open-Source Software Lexicons

by Derrin Pierret and Denys Poshhyvanyk
Introduction

- Lexicon: vocabulary used in a program
  - identifiers, keywords, symbols, etc.

- Lexicon metrics could distinguish programs

Recent Studies on Power Laws

<table>
<thead>
<tr>
<th>Study</th>
<th>Analysis</th>
<th>Statistics</th>
<th>Findings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our breadth study</td>
<td>yes</td>
<td>no</td>
<td>12 142</td>
</tr>
<tr>
<td>Zhang [1]</td>
<td>yes</td>
<td>no</td>
<td>Java 12</td>
</tr>
<tr>
<td>Concas et al. [2]</td>
<td>no</td>
<td>yes</td>
<td>3 3</td>
</tr>
<tr>
<td>Baxter et al. [3]</td>
<td>no</td>
<td>yes</td>
<td>Java 56</td>
</tr>
<tr>
<td>Louridas et al. [5, 9]</td>
<td>no</td>
<td>yes</td>
<td>6 19</td>
</tr>
</tbody>
</table>
Background: Zipf-Mandelbrot Law

\[ f = \frac{C}{(r + \beta)^\alpha} \]

Where:
- \( f \) = word frequency
- \( r \) = word rank
  (1st most common, 2nd, 3rd, etc)
- \( \alpha, \beta, C \) = constant for fitting

• Research Questions
  - Is equation reliable?
  - What do constants reflect?
Goals

How well does Zipf's Law fit...

• token distributions amongst...
  o projects
  o languages
  o paradigms

• word distributions amongst...
  o documentation
  o bug reports
Case Study Design

4 Paradigms, 3 Languages each:

- **Object Oriented:** Java, C++, Smalltalk
- **Imperative:** Matlab, C, PHP
- **Markup:** HTML, XML, TeX
- **Functional:** Haskell, Scheme, OCaml

Roughly 10 programs per language

141 programs total, 9 non-program artifacts
Results

Sample of projects with gathered statistics

<table>
<thead>
<tr>
<th>Proj Name</th>
<th>Lang/Artifact</th>
<th>Paradigm</th>
<th>a</th>
<th>β</th>
<th>C</th>
<th>MMRE</th>
<th>Avg Token Length</th>
<th>LOC</th>
<th>Voc Size</th>
<th>Proj Size (Tokens)</th>
<th>Est Proj Size</th>
<th>Est MMRE</th>
<th>Developers</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>aMule</td>
<td>C++</td>
<td>OOP</td>
<td>1.31</td>
<td>1.00</td>
<td>589,252.14</td>
<td>0.13</td>
<td>9.82</td>
<td>123,830</td>
<td>27,008</td>
<td>1,064,595</td>
<td>1,439,936</td>
<td>0.35</td>
<td>23</td>
<td>File Sharing</td>
</tr>
<tr>
<td>impresscms</td>
<td>PHP</td>
<td>Proc</td>
<td>1.38</td>
<td>3.00</td>
<td>476,889.87</td>
<td>0.11</td>
<td>9.25</td>
<td>70,782</td>
<td>13,451</td>
<td>741,400</td>
<td>698,712</td>
<td>0.06</td>
<td>53</td>
<td>Internet</td>
</tr>
<tr>
<td>liquidsoap</td>
<td>OCaml</td>
<td>Func</td>
<td>1.44</td>
<td>4.80</td>
<td>284,331.09</td>
<td>0.13</td>
<td>8.23</td>
<td>29,589</td>
<td>6,761</td>
<td>233,358</td>
<td>282,452</td>
<td>0.21</td>
<td>11</td>
<td>Multimedia</td>
</tr>
</tbody>
</table>

"Good Fit": Mean Magnitude of Relative Error (MMRE) < 0.25
  
  o MRE: % of the actual value that the estimate is off by
    ▪ | actual = 10,000 | estimate = 8,000 |
    ▪ | 2000 = 10,000 * 0.2 | MRE = 0.2 |
  
  o MRE collected for each token in program
  o Project MMRE is average of the token MREs

Most projects met criteria for "good fit"
Good Fit (MMRE = 0.11)
Bad Fit (MMRE = 0.32)
## Average MMRE Values

<table>
<thead>
<tr>
<th>Category</th>
<th>Java</th>
<th>C++</th>
<th>Smalltalk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Oriented</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.18</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td><strong>Imperative</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.18</td>
<td>0.17</td>
<td>0.17</td>
</tr>
<tr>
<td><strong>Markup</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.21</td>
<td>0.19</td>
<td>0.15</td>
</tr>
<tr>
<td><strong>Functional</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.16</td>
<td>0.16</td>
<td>0.17</td>
</tr>
<tr>
<td><strong>Non-Source</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>0.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bug Reports</td>
<td>0.23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Revised Program Length Estimation

• Halstead: Software Science equation
  o Links vocabulary size with program size

• Zhang: Saw inaccuracy with Halstead on Java code
  o Saw Zipf-Mandelbrot work for Java tokens
  o Devised new length estimation equation from Zipf

\[ N^\wedge = C \frac{(n + \beta)^{1-\alpha} - (1 + \beta)^{1-\alpha}}{1 - \alpha} \]

Where:
- \( N^\wedge \) = program length
- \( n \) = vocabulary length
- \( \alpha, \beta, C \) = Zipf constants

• Estimations were only sometimes accurate
  o Good estimation for 84 out of 141 projects (60%)
Conclusions

Zipf's Law is able to describe token distributions well for:
- Project Source, Languages, Paradigms
- Documentation, Bug Reports

Program size estimation equation needs further investigation

Questions left to explore:
- Meaning behind Zipf goodness of fit or generated constants
- Where types of tokens appear in distribution

Online Appendix:

http://www.cs.wm.edu/~dpierret/zipf-appendix.html