On the Use of Stemming for Concern Location and Bug Localization in Java

Emily Hill

Shivani Rao, Avinash Kak

Montclair State University
Purdue University
Problem: Source Code Search

- Query: “add item”
  - Stemming used to strip suffixes & improve recall by reducing words to root form, or stem
    - Widely studied in Information Retrieval (IR)
    - Not so much for SE (very different document style)
A Brief History of Stemming

• **Light Stemmers** (tend not to stem across parts of speech)
  
  – **Porter** (1980): algorithmic (rule-based), simple & efficient
    
    • Most popular stemmer in IR & SE
    • **Snowball** (2001): minor rule improvements
  
  – **KStem** (1993): morphology-based
    
    • based on word’s structure & hand-tuned dictionary
    • in experiments shown to outperform porter’s

• **Heavy Stemmers** (can overstem, reducing precision)
  
  – **Lovins** (1968): algorithmic
  
  – **Paice** (1990): algorithmic
  
  – **MStem**: morphological (PC-Kimmo), specialized for source code using word frequencies
Our Contribution

Investigate use of stemming for 2 different types of Java source code search tasks with various queries:

• **Bug Localization:** find methods in bug fix (IR: Unigram Model)
  
  291 bugs from iBugs dataset (ASPECTJ) with queries:
  
  – **M291:** all 291 bugs, with initial bug description as query (not title)
  – **Medium:** initial bug description of 126 bugs that contain both title & comments (not much code)
  – **Short:** title of 126 bugs  – **Long:** title + full comments of 126 (some code)

• **Concern Location:** find methods implementing a concept of interest, given keyword-style queries (IR: tf-idf)
  
  – 8 action-oriented concerns from 4 programs (AOC), 48 queries
  – 215 documentation-based concerns from Rhino (Rhino), 645 queries
Analysis Methodology

- **MAP** (Mean Average Precision): $\text{AP} = \text{average precision at each relevant result}$

- **Rank Measure** [Hull ’96]: rank of relevant documents for each query

- **Qsets** [Hull ’96]: partition queries into sets:
  - $Q_+$: stemming helps
  - $Q_-$: stemming hurts
  - $Q_{=}$: stemming has no effect
  - $Q_{\text{vary}}$: effect depends on stemmer
Results: Bug Localization

### Mean MAP Difference Scores

<table>
<thead>
<tr>
<th>Query</th>
<th>Paice</th>
<th>KStem</th>
<th>MStem</th>
<th>Snowball</th>
<th>Porter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long</td>
<td>-0.02806</td>
<td>-0.006090</td>
<td>-0.004199</td>
<td>-0.008529</td>
<td>-0.01055</td>
</tr>
<tr>
<td>Medium</td>
<td>0.002638</td>
<td>0.007970</td>
<td>0.01400</td>
<td>-0.003243</td>
<td>0.007280</td>
</tr>
<tr>
<td>M291</td>
<td>-0.006703</td>
<td>-0.002056</td>
<td>-0.003738</td>
<td>-0.008391</td>
<td>-0.002462</td>
</tr>
<tr>
<td>Short</td>
<td>-0.008479</td>
<td>0.003343</td>
<td>0.0004492</td>
<td>0.0003093</td>
<td>-0.002758</td>
</tr>
</tbody>
</table>

• Stemming plays more of a role for shorter queries
Results: Concern Location

**Mean MAP Difference Scores**

<table>
<thead>
<tr>
<th>Set</th>
<th>Paice</th>
<th>KStem</th>
<th>MStem</th>
<th>Snowball</th>
<th>Porter</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOC</td>
<td>0.03072</td>
<td>0.02619</td>
<td>0.02548</td>
<td>0.01576</td>
<td>0.01762</td>
</tr>
<tr>
<td>Rhino</td>
<td>0.002955</td>
<td>0.0007937</td>
<td>-0.0008919</td>
<td>-0.0001163</td>
<td>-0.0001374</td>
</tr>
</tbody>
</table>

- **Paice** significantly outperforms other stemmers for Rhino, points to possible interaction with tf-idf
Results: Qsets

Number of queries where stemming helps ($Q_+$), hurts ($Q_-$), has no effect ($Q_=$), and where performance varies ($Q_{vary}$).

**Bug Localization**  Mean number of words in query, notes

<table>
<thead>
<tr>
<th>Query Type</th>
<th>$Q_+$</th>
<th>$Q_-$</th>
<th>$Q_=$</th>
<th>$Q_{vary}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short 8.5</td>
<td>29</td>
<td>46</td>
<td>4</td>
<td>47</td>
</tr>
<tr>
<td>Medium 247</td>
<td>25</td>
<td>34</td>
<td>6</td>
<td>61</td>
</tr>
<tr>
<td>M291 320</td>
<td>53</td>
<td>92</td>
<td>12</td>
<td>134</td>
</tr>
<tr>
<td>Long 770, code</td>
<td>25</td>
<td>36</td>
<td>5</td>
<td>60</td>
</tr>
</tbody>
</table>

**Concern Location**

<table>
<thead>
<tr>
<th>Query Type</th>
<th>$Q_+$</th>
<th>$Q_-$</th>
<th>$Q_=$</th>
<th>$Q_{vary}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AOC 2, verbs</td>
<td>18</td>
<td>9</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Rhino 4, nouns</td>
<td>112</td>
<td>239</td>
<td>70</td>
<td>224</td>
</tr>
</tbody>
</table>
Conclusion & Discussion

• So far, success of any particular stemmer situation dependent (we can’t yet generalize)
  – Stemmer success seems dependent on query nature & retrieval model

• Are there other variables missing from our model of the problem, or is this due to the nature of stemming/searching itself?
  – Query length, presence of code/identifiers
  – Query difficulty (how well matches code words)

• Future Work: explore the interaction between retrieval model, query length/type, & stemmer