Source Code Identifier Splitting Using Yahoo Image and Web Search Engine

Ashish Sureka

Faculty Member at IIIT-D (ashish@iiitd.ac.in)
Source code identifiers

- Names of classes
- Names of interfaces
- Methods or Functions Names
- Names of Variable
- Names of Formal Parameters or Arguments

Sequence of characters (a string), Consisting of one or many tokens

Identifier Tokens or Terms

- Word (a word in a standard English dictionary)
- Well-known and commonly used acronyms (country USA or the company IBM)
- Short-forms like Mr. or max
- Domain-specific abbreviations (such as str for string, pntr for pointer, rect for rectangle)
Technical Problem: **Identifier Splitting** OR **Identifier Tokenization**

**Application of Identifier Splitting**

- Feature or concept location in source-code
- Automatic traceability link recovery between software artifacts
Research Motivation and Aims  Challenges: Source Code Identifier Splitting

Technical Challenge(s)

- Lack of explicit markers or boundaries between various component words

submitconfpaper  submit_conf_paper  submitConfPaper

Research Problem

concept boundaries are explicit or known in advance

- Significant percentage of identifiers in source-code do not contain explicit markers between tokens [Butler, 2011]

- Approximately **15%** of the identifiers in the sample dataset are non-trivial to tokenize because of lack of explicit markers [Butler, 2011]

Examples from [Butler, 2011]

- setOSTypes (set, OS, Types)
- hasSV UID (has, SVU, ID)
- DAYSforMONTH (DAYS, for, MONTH)
- ALTORENDSTATE (ALT, OR, END, STATE)
- isOSGiCompatible (is, OSGi, Compatible)
- maxprefwidth (max, pref, width)

Examples from [Feild, 2006]

- rcvptr (rcv, ptr)
- trustdom (trust, dom)
- listlength (list, length)
- sockaddr (sock, addr)
Research Motivation and Aims

Broad and Specific Research Objectives

- To investigate novel approaches to address the problem of automatic identifier splitting
- To examine approaches that is complementary to traditional methods
- To investigate the application of *Yahoo image search (based on visual domain and image features)* for the task of automatic identifier splitting
- To investigate the application of *Yahoo web search (free-form textual domain and linguistic features)* for the task of automatic identifier splitting
Related Work and Research Contributions

Three lines of research

- Automatic identifier splitting
- Automatic expansion of abbreviations in identifier names
- Exploiting web-search engine results for problem-solving in various domains
## Related Work and Research Contributions

### Previous approaches for source-code identifier splitting

<table>
<thead>
<tr>
<th>Study</th>
<th>Main Idea</th>
<th>Reference Database</th>
<th>Evaluation Dataset</th>
<th>Algorithm Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field et al. (2006)</td>
<td>Longest substring search based on greedy algorithm</td>
<td>Linux spell checker Is-Pell</td>
<td>4,000 identifiers randomly chosen from C, C++, Java, Fortran Programs</td>
<td>Recursive algorithm</td>
</tr>
<tr>
<td>Field et al. (2006)</td>
<td>Artificial Neural Network (ANN), Input: Character sequence, Output: Split position</td>
<td>-</td>
<td>4,000 identifiers randomly chosen from C, C++, Java, Fortran Programs</td>
<td>Requires training (model building)</td>
</tr>
<tr>
<td>Enslen et al. (2009)</td>
<td>Mining word frequencies in source code</td>
<td></td>
<td>8000 identifiers from open source Java programs</td>
<td>Recursive algorithm, Scoring function for split decision</td>
</tr>
<tr>
<td>Lawrie et al. (2010,11)</td>
<td>Vocabulary normalization, Generate and test algorithm, co-occurrence computation</td>
<td>Google Web 1T 5-gram database</td>
<td>487 identifiers from which-2.20 a2ps</td>
<td>Normalizes source code vocabulary, Performs identifier expansion</td>
</tr>
<tr>
<td>Butler et al. (2011)</td>
<td>Greedy algorithm, Forward and backward tokenization pass</td>
<td>Published word lists with 117,000 entries</td>
<td>28,000 identifier names drawn from 60 open source Java projects</td>
<td>Addresses identifier names containing digits</td>
</tr>
</tbody>
</table>
Abbreviations Expansion in Identifier Names

nummsg
- *num* is an abbreviation (short-form) of number (long-form)
- *msg* is an abbreviation of message

strlen
- *str* is an abbreviation of string
- *len* is an abbreviation of length

Abbreviation-Expansion: mapping *num* to number and *msg* to message

Related Papers [Hill, 2008] [Lawrie, 2010] [Lawrie, 2011]

Exploiting Page-Hits and Search-Results

- Web-search engines (in particular, search results returned by the retrieval engines) as tools
- Exploiting vast amount of data (a corpus) on the Web as an external knowledge-base
- Using search engine page hits as a proxy for n-gram frequencies [Nakov, 2005]
- Extracting the meaning of words and phrases from the world-wide-web using Google page counts [Cilibrasi, 2007]
- Using page count and snippets in retrieved results to do perform word semantic similarity measurement [Lu, 2010]
- Propose a metric called as Flickr Distance (FD) to measure the visual correlation between concepts [Wu, 2011]
Comparison with Previous Approaches and Novel Contributions

➢ Our technique exploits *Yahoo Image Search* which is a unique feature in context to traditional methods.

➢ Computing conceptual correlation between two candidate terms using the *visual domain* (con-currence of objects representing the terms in a image) is a new strategy.

➢ Previous approaches uses a look-up table like a English dictionary, lexical database behind a spell checker and Google Web 1T 5-gram database - and we use the web-pages and images indexed by a *general purpose search engine like Yahoo*.

Novel Contributions

➢ A method for source-code or program identifier splitting based on computing *page-hits* or *number of search engine results* returned by *Yahoo web* and *image search engine*.

➢ An empirical analysis and performance evaluation of the proposed method on real-world dataset.
Most complex case of identifier splitting: all the letters in *same-case* (all lower-case or all upper-case) and identifiers that contain *digits*

Next Slides Proposed solution approach: we do not exploit case or capitalization information
Yahoo Image or Web Search Based Token Splitting Score (**SplitScore**)

**Input:** $S_{\text{LEFT}}$, $S_{\text{RIGHT}}$, $b$, $k$

**Output:** $\text{SCORE}$

1: $f_{\text{LEFT}} \leftarrow \log_b[HITS(S_{\text{LEFT}})]$
2: $f_{\text{RIGHT}} \leftarrow \log_b[HITS(S_{\text{RIGHT}})]$
3: $f_{\text{LEFT} \land \text{RIGHT}} \leftarrow \log_b[HITS(S_{\text{LEFT} \land \text{RIGHT}})]$
4: $l_{\text{LEFT}} \leftarrow \text{LENGTH}(S_{\text{LEFT}})$
5: $l_{\text{RIGHT}} \leftarrow \text{LENGTH}(S_{\text{RIGHT}})$
6: $l_{\text{MIN}} \leftarrow \text{MIN}(l_{\text{LEFT}}, l_{\text{RIGHT}})$
7: $\text{SCORE} \leftarrow f_{\text{LEFT}} \times f_{\text{RIGHT}} \times f_{\text{LEFT} \land \text{RIGHT}} \times (l_{\text{MIN}})^k$

Compute $\text{SCORE}$ for the SPLIT POSITION

$S_{\text{LEFT}} = AB$
$S_{\text{RIGHT}} = CDE$

$S_{\text{LEFT}} = ABC$
$S_{\text{RIGHT}} = DE$

$f_{\text{LEFT}}$, $f_{\text{RIGHT}}$ = function of number of search results

**SPLIT SCORE:** $F(\text{number of search results for } S_{\text{LEFT}} \text{ and } S_{\text{RIGHT}}, \text{ length of } S_{\text{LEFT}} \text{ and } S_{\text{RIGHT}})$
Token Split Decision Function Returning True for False (SplitDecision)

\[ \text{Decide: TO SPLIT OR NOT} \]

Input: \( S, S_{LEFT}, S_{RIGHT}, k \)

Output: \( \text{DECISION} \)

1: \( f \leftarrow \text{HITS}(S) \)
2: \( f_{LEFT} \leftarrow \text{HITS}(S_{LEFT}) \)
3: \( f_{RIGHT} \leftarrow \text{HITS}(S_{RIGHT}) \)
4: \( l \leftarrow \text{LENGTH}(S) \)
5: \( l_{LEFT} \leftarrow \text{LENGTH}(S_{LEFT}) \)
6: \( l_{RIGHT} \leftarrow \text{LENGTH}(S_{RIGHT}) \)
7: \( \text{SCORE} \leftarrow f \times (l)^k \)
8: \( \text{SCORE}_{LEFT} \leftarrow f_{LEFT} \times (l_{LEFT})^k \)
9: \( \text{SCORE}_{RIGHT} \leftarrow f_{RIGHT} \times (l_{RIGHT})^k \)
10: \( \text{if } (\text{SCORE}_{LEFT} \geq \text{SCORE}) \land (\text{SCORE}_{RIGHT} \geq \text{SCORE}) \text{ then} \)
11: \( \text{DECISION} \leftarrow \text{True} \)
12: \( \text{else} \)
13: \( \text{DECISION} \leftarrow \text{False} \)
14: \( \text{end if} \)

\( \text{SCORE} \) is a function of number of search results \( (f) \) & string length \( (l) \)
Identifier Name Splitting into Token(s) \((\text{SplitIdentifierName})\)

Compute **SplitScore** for ALL Split Positions

Record the split position with **MAX SPLIT SCORE**

**Recursive Algorithm**

IF DECISION = TRUE then compute call SplitIdentifierName for \(S_{\text{LEFT}}\) and \(S_{\text{RIGHT}}\)
Application of Number of Search Results on Identifier Splitting Algorithm

The prevalence of the concept in images (Yahoo Image Search) and text (Yahoo Web Search) indexed by a web search engine is reflected in the number search results or hits.

- The number of hits for the term conference is more than the term conferen.

- Number of search results is used as a proxy to check if the concept represented by a string exists in the real world and to what extent.
The co-location or co-occurrence frequency of two concepts is determined by invoking the search engine using an and operator.

For identifier \textit{conferencepaper} then the degree of co-occurrence of concepts \textit{conference} and \textit{paper} is determined by invoking the search engine with a query consisting of both the terms using an and operator.

The higher the co-location (text search) and co-occurrence (image search) frequency, the higher the split score.
Visual-Domain Based Search

Main components of the proposed solution

- Likelihood of a string (represented by a token) representing a real-world concept
- Measuring semantic relatedness between any two given concepts (to decide split positions)

Several types of conceptual correlations such as synonymy, visual similarity, meronomy and concurrency and measuring conceptual similarity using visual information is more coherent to human cognition than text-based distance [Wu, 2008, 2011]

- Meronomy (car-wheel, building-window, tree-leaf) and concurrence (airplane-airport, desk-chair, sky-cloud) are better captured by measuring frequency of co-occurrence of the two given concepts based on visual content of an image [Wu, 2008, 2011]
Number of images returned for the term camera and phone is large which increases the likelihood of the split position (camera and phone) being recommended by the program.

Figure demonstrates that a general purpose image search engine retrieves images of an HTML Editor when entering the search string as HTML and Editor where the term and is a search operator and this feature can be exploited for the task of identifier splitting.
Performance Evaluation and Empirical Results

Method to compute success and effectiveness of the approach

- Input to Algorithm: Identifier consisting of an arbitrary number of constituent terms
- The ground-truth (the actual result or answer-key) is already known
- Performance measurement: comparing the output produced by the approach with the available ground-truth

Yahoo! Search BOSS API [http://developer.yahoo.com/search/boss/]

- Open search and a data service platform
- Programmatically retrieving web and image search results from Yahoo search engine
- API provides services - takes search query as input (with various operators such as AND, OR)
- Returns the total number of search results for the given query
Evaluation Dataset

**RPME (Research Papers Motivating Examples)**

- We extract motivating examples of identifiers used in four research papers on the topic of automatic identifier splitting


- These four research papers contain several identifiers used to illustrate the technical challenges in the problem of automatic identifier splitting

- These examples are considered as difficult cases for an identifier splitting method by the respective authors

- We extract all such identifiers from the papers and package them into an evaluation dataset called as RPME

- Examples contain a variety of identifier types: identifiers containing short-forms or abbreviations, digits and long strings having multiple constituent terms
Evaluation Dataset

**BT11 Dataset [Butler 2011]**

- Contains a variety of identifier types extracted from source-code elements such as class names, method names, field names and formal arguments.

- Extracted from a database of 827,475 unique identifier names from 16.5 MSLOC of Java from 60 projects [Butler 2011].
### Performance Evaluation an Empirical Results

#### Accuracy Results [RPME dataset]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>CI</th>
<th>PCI</th>
<th>CW</th>
<th>PCW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feild et al. 2006</td>
<td>5</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Enslen et al. 2009</td>
<td>20</td>
<td>50%</td>
<td>25%</td>
<td>55%</td>
<td>25%</td>
</tr>
<tr>
<td>Madani et al. 2010</td>
<td>17</td>
<td>70%</td>
<td>6%</td>
<td>59%</td>
<td>23%</td>
</tr>
<tr>
<td>Butler et al. 2011</td>
<td>17</td>
<td>47%</td>
<td>35%</td>
<td>35%</td>
<td>53%</td>
</tr>
</tbody>
</table>

- **CI**: correct result using image search engine
- **PCI**: partially correct result (indicate that in a multiword identifier some of the constituents words are correctly tokenized but not all) using image search engine
- **CI**: correct result using web search engine
- **PCW**: partially correct result (web-search engine)
### Performance Evaluation an Empirical Results

#### Accuracy Results [BT11 dataset]

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>CI</th>
<th>PCI</th>
<th>CW</th>
<th>PCW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Names</td>
<td>100</td>
<td>33%</td>
<td>34%</td>
<td>34%</td>
<td>17%</td>
</tr>
<tr>
<td>Method Names</td>
<td>100</td>
<td>37%</td>
<td>23%</td>
<td>37%</td>
<td>12%</td>
</tr>
<tr>
<td>Field Names</td>
<td>100</td>
<td>32%</td>
<td>30%</td>
<td>38%</td>
<td>25%</td>
</tr>
<tr>
<td>Formal Arguments</td>
<td>100</td>
<td>31%</td>
<td>33%</td>
<td>34%</td>
<td>37%</td>
</tr>
<tr>
<td>Local Variable</td>
<td>100</td>
<td>31%</td>
<td>24%</td>
<td>36%</td>
<td>31%</td>
</tr>
</tbody>
</table>

**CI** : correct result using image search engine

**PCI** : partially correct result (indicate that in a multiword identifier some of the constituents words are correctly tokenized but not all) using image search engine

**CI** : correct result using web search engine

**PCW** : partially correct result (web-search engine)
Successful Examples and Solution Advantages

- Can handle non-dictionary terms and does not require any pre-annotated dataset

- Able to correctly split identifiers containing terms like ptr (pointer), cnt (count), rect (rectangle), lbl (label), addr (address), lib (library) and idx (index)

- Able to successfully extract concepts from long identifier names like weblogic81databaseconverter containing many concepts and digits

- Able to successfully split the identifier awtlockaccess into awt, lock and access which contains a domain specific abbreviation AWT

- Identifiers like getIPv6, ARROWMBR and brandAIX having mixed-case, containing digits, domain specific non-dictionary abbreviations were correctly split into its constituents terms
Summary

- Present method to automatically split multi-term source-code identifiers into its constituent concepts
- Using a recursive algorithm invoking a general purpose image and web search engine
- Exploit the vast amount of text documents and images indexed by general purpose search engines as external knowledge-base
- External knowledge-base For determining the optimal split position for a given character sequence or string
- Investigate the usefulness of both visual and textual domain and present empirical evidences supporting the proposed hypothesis