Towards an Intelligent Code Search Engine

Jinhan Kim, Sanghoon Lee, Seung-won Hwang, Sunghun Kim

Ioannis Gkermpeziotis
M-2010-1131
grad1131@di.uoa.gr
Paper Outline

1. Introduction
2. Our Approach
   • Summarization
   • Representation
   • Diversification
   • Ranking
3. Evaluation
   • Comparison with JavaDocs
   • Comparison with Code Search Engines
   • Comparison with Golden Standard
   • Java Developer Feedback
4. Case Study
   • Study Design
   • Participants
   • Study Result
5. Conclusion
6. Acknowledgments
Reusing existing Application Programming Interfaces (APIs)
• Searching for specific API on the Web
  • API documents
  • API usage examples
• Browsing different APIs to select the appropriate one
  • API description
Introduction

Google Code (2009)

Koders (2009)
Introduction

Existing search engines
• use API name as a query keyword

Search results
• matches in the comments of source codes
• fail to provide usage examples

*Koders Top-2 results when the query is “Connection prepareStatement”*
Introduction

Solutions:

Rich set of examples (MSDN, Leopard Reference Library)
  • time consuming

Search engines code results implementation
  • low quality

Code recommendation approach
  • complex contexts
eXoaDocs

An intelligent code search engine that searches, summarizes, and embeds the necessary information in advance by automatically augmented with high-quality code, diverse examples.

An example page of generated eXoaDocs. (a) The popularity information of methods (b) Code examples (2 - 5) (c) User’s feedback button
Approach

Main Modules
- Summarization
- Representation
- Diversification
- Ranking
Summarization

Summarize the resulted codes into good example snippets
  • Actual API should be included
  • Irrelevant code, regardless textual proximity, should be excluded

Method Extraction
  Identify methods that contain given API (usage, context)

API slicing
  Extract only semantically relevant lines for the given API
  • Declaring input arguments
  • Changing input arguments values
  • API class declaration
  • API calls
Summarization

- eXoaDocs code example of “Character.hashcode()”

```java
public void test (TestHarness harness){
    Character a = new Character ("uffda");
    Character b = new Character ("Z");
    harness.check (a.hashCode(), 65498);
    harness.check (b.hashCode(), 90);
}
```

- Abstract SyntaxTree

- Intra-method Analysis Table

<table>
<thead>
<tr>
<th>Line</th>
<th>Class</th>
<th>API name</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>a : Character</td>
<td>hashCode</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>b : Character</td>
<td>hashCode</td>
<td></td>
</tr>
</tbody>
</table>
Representation

Representation of code example for further analysis

Simple texts
+ Highly efficient
- Neglects semantic context

ASTs
+ Considers semantic context
- Expensive

Element Vectors
Inspired by a clone detection algorithm (DECKARD)
Computing similarity using vector comparison
Diversification

Diversify code examples to identify different usage types
Clustering based on extracted vectors (2 - 5 clusters)

Invoke k-means four times and select result according to:

- **Centroid distribution**: Clustering where centroids are distributed evenly (not skewed). $1/var_i, k = i$

- **Sum of squared error (SSE)**: Clustering with the smallest SSE suggests that centroids are statistically better representatives of all vectors in the cluster and thus indicates high quality clustering results. Difference of SSE $\Delta_i, k = i, i - 1$

- **Hierarchical clustering**: Usage type clusters typically represent a hierarchical structure. 1 if all members of cluster for $k = i$ come from same cluster for $k = i - 1$
**Intra-cluster ranking:** select one distinctive and representative code example from each cluster

- **Representativeness:** L1 distance from centroid. \[ \min(1/\text{similar}, 1) \]
- **Conciseness:** those with fewer lines of code, are highly ranked. \( 1/\text{length} \)
- **Correctness:** right class or matching of arguments.

**Inter-cluster ranking:** rank the representative code examples

- Count of example codes in each cluster quantify popularity
Evaluation

1. Comparison with JavaDocs (20,000 methods)
   - eXoaDocs augmented examples for more than 75%
   - JavaDocs examples for only 2%

2. Comparison with Code Search Engines
   - Koders and Google Code search results irrelevant (70% and 78%), including comments, imports etc.
   - proper usage for eXoaDocs 92%, while Koders 22% and Google Code 12%
   - eXoaDocs duplicates 8.7%, while among relevant examples Koders had 30.8% and Google Code 16.7%

3. Comparison with Golden Standard
   - Results of 66% precision and 60% recall
   - Summarization 82% precision and 73% recall

4. Java Developer Feedback
Case Study

Study Design

Subjects: 24 undergraduate students
Assignment: build SQL applications using java.sql package

Subjects were randomly divided into two groups to use either eXoaDocs or JavaDocs, to complete the tasks:

- Task$_1$: Establish a connection to database
- Task$_2$: Create SQL statements
- Task$_3$: Execute SQL statements
- Task$_4$: Present query results

Productivity

- Overall development time
- API document lookups

Code Quality
Case Study

Study Result

Productivity

Average completion time

<table>
<thead>
<tr>
<th>Group</th>
<th>Task(_1)</th>
<th>Task(_2)</th>
<th>Task(_3) &amp; Task(_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>eXoaGroup</td>
<td>8:53</td>
<td>23:34</td>
<td>30:32</td>
</tr>
<tr>
<td>JDocGroup</td>
<td>14:40</td>
<td>25:03</td>
<td>32:03</td>
</tr>
</tbody>
</table>

Average document lookups

<table>
<thead>
<tr>
<th>Group</th>
<th>Total lookups</th>
<th>Distinct lookups</th>
<th>Relevant lookups</th>
<th>relevant distinct</th>
</tr>
</thead>
<tbody>
<tr>
<td>eXoaGroup</td>
<td>5.67</td>
<td>3.25</td>
<td>2.33</td>
<td>0.72</td>
</tr>
<tr>
<td>JDocGroup</td>
<td>17.58</td>
<td>7.5</td>
<td>3.25</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Code Quality

Number of subjects passing test suite

<table>
<thead>
<tr>
<th>Group</th>
<th>Task(_1)</th>
<th>Task(_2)</th>
<th>Task(_3)</th>
<th>Task(_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>eXoaGroup</td>
<td>11</td>
<td>11</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>JDocGroup</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Thank you!

Questions?