Inferring semantically related words from software context

Jinqiu Yang, Lin Tan
University of Waterloo
Motivation

I need to find all functions that disable interrupts in the Linux kernel. Hmm, so I search for “disable*interrupt”.

**MISSING:**
disable_irq(...), mask_irq(...)

**New Search Queries:**
“disable*irq”, “mask*irq”

BUT how am I supposed to know???
How to Find Synonyms or Related Words?

Can’t find that disable & mask are synonyms!

Guess on my own
Ask developers
Our Approach: Leveraging Context

• Comments:

  “Disable all interrupt sources”

  “Disable all irq sources”

• Identifiers:

  void mask_all_interruptions()

  void disable_all_interruptions()

Real comments and identifiers from the Linux kernel

• We call a pair of such semantically related words an rPair.
Contributions

• A general context-based approach to automatically infer semantically related words from software context

• Has a reasonable accuracy in 7 large code bases written in C and Java.

• Is more helpful to code search than the state of art.
Outline

• Motivation, Intuition and Contributions
• Our Approach
  • A Running Example: Parsing, Clustering, Extracting, Refining
• Evaluation Methods & Results
• Related Work
• Conclusion
A Running Example

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

Real comments from Apache HTTPD Server
Extracting rPairs

\[ \text{SimilarityMeasure} = \frac{\text{Number of Common Words in the Two Sequences}}{\text{Total Number of Words in the Shorter Sequence}} \]

threshold = 0.7

You can find how different thresholds affect our results in our paper.
Running Out of Time

• Pairwise comparisons of a large number of sequences is expensive.

• 519,168 unique comments in the Linux kernel $\rightarrow$ over 100 billion comparisons
Clustering

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file
Clustering

maybe add a higher-level description

min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file
Clustering

maybe add a higher-level description
return err maybe add a higher-level desc

maybe add a higher-level description
min of spare daemons
data in the appropriate order
the compiled max daemons
an iovec to store the trailer sent after the file
data in the wrong order
an iovec to store the headers sent before the file
return err maybe add a higher-level desc
if a user manually creates a data file

data
data in the appropriate order
data in the wrong order
if a user manually create a data file

iovec
an iovec to store the headers sent before the file
an iovec to store the trailer sent after the file
Pairwise comparisons of a large number of sequences is expensive.

519,168 unique comments in the Linux kernel → over 100 billion comparisons.

Clustering speeds up the process for the Linux kernel by almost 100 times.
Refining rPairs

- **Filtering:**
  - Using stemming to remove rPairs that consists of words with the same root, e.g., (called, call).

- **Normalization:**
  - (threads, daemons) ⟷ (thread, daemon).
  - (called, invoked) ⟷ (call, invoke)
Outline

• Motivation, Intuition and Contributions
• Our Approach
  • A Running Example: Parsing, Clustering, Extracting, Refining
• Evaluation Methods & Results
• Related Work
• Conclusion
Evaluation Methods

• **Extraction Accuracy**
  • 7 large code bases, in Java & C, from Comment-Comment, Code-Code, Comment-Code

• **Search-Related Evaluation**
  • Comparison with SWUM [Hill Phd Thesis] in Code-Code
## Comment-Comment Accuracy Results

<table>
<thead>
<tr>
<th>Software</th>
<th>rPairs</th>
<th>Accuracy</th>
<th>Not in Webster or WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>108,571</td>
<td>47%</td>
<td>76.6%</td>
</tr>
<tr>
<td>HTTPD</td>
<td>1,428</td>
<td>47%</td>
<td>93.6%</td>
</tr>
<tr>
<td>Collections</td>
<td>469</td>
<td>74%</td>
<td>97.3%</td>
</tr>
<tr>
<td>iReport</td>
<td>878</td>
<td>84%</td>
<td>95.2%</td>
</tr>
<tr>
<td>jBidWatcher</td>
<td>111</td>
<td>64%</td>
<td>98.4%</td>
</tr>
<tr>
<td>javaHMO</td>
<td>144</td>
<td>56%</td>
<td>91.1%</td>
</tr>
<tr>
<td>jajuk</td>
<td>203</td>
<td>69%</td>
<td>94.2%</td>
</tr>
<tr>
<td><strong>Total/Average</strong></td>
<td><strong>111,804</strong></td>
<td><strong>63%</strong></td>
<td><strong>91.7%</strong></td>
</tr>
</tbody>
</table>

We randomly sample 100 rPairs per project for manual verification (all 111 for jBidWatcher).

- The majority (91.7%) of correct rPairs discovered are not in Webster or WordNet.
Evaluation Methods

• **Extraction Accuracy**
  • 7 large code bases, in Java & C, from Comment-Comment, Code-Code, Comment-Code

• **Search-Related Evaluation**
  • Comparison with SWUM [Hill Phd Thesis] in Code-Code
Search-Related Evaluation

In jBidWatcher, “Add auction”

Query expansion: “XXX auction”

Our approach

new register ...

SWUM

register ...

17
Search-Related Evaluation

In jBidWatcher, "Add auction"

```
JBidMouse.DoAuction(...)
AuctionServer.registerAuction(...)
AuctionManager.newAuctionEntry(...)
FilterManager.addAuction(...)
...
```

add ➔ register, do, new

SWUM gold set

our gold set
Search-Related Evaluation

In jBidWatcher, “Add auction”

\[
\text{add} \rightarrow \text{register, do, new}
\]

Our approach (55 words)

- new
- register
- do
- load
- ...

Precision = \(3/55 = 5.5\%\)
Recall = \(3/3 = 100\%\)

SWUM (84 words)

- register
- do
- ...

Precision = \(2/84 = 2.3\%\)
Recall = \(2/3 = 67.7\%\)
Search-Related Evaluation

In jBidWatcher, “Add auction”

`add -> register, do, new`

---

Our approach

55 words

SWUM

84 words

Our approach achieves higher precision and higher/equal recall for **5** out of **6** rPair groups in the gold set.

```
load ...
```

Precision = 3/55 = 5.5%
Recall = 3/3 = 100%

```
...
```

Precision = 2/84 = 2.3%
Recall = 2/3 = 67.7%
Related Work

- **Verb-DO (Direct Object)** [Shepherd et al. AOSD] & **SWUM** - Improved version of Verb-DO [Hill Phd Thesis]

- Requires Natural Language Processing (NLP) techniques
- Requires manually generated heuristics
Conclusions

• A **simple, general** technique to automatically infer semantically related words from software context
• No Natural Language Processing (NLP) required
• Reasonable accuracy in 7 large C & Java code bases
• The majority of rPairs discovered are not in the dictionaries or WordNet.
• Higher precision & recall than the state of art