On the Use of Relevance Feedback in IR-Based Concept Location

Gregory Gay*, Sonia Haiduc**, Andrian Marcus**, Tim Menzies*

* West Virginia University, Morgantown, WV, USA
** Wayne State University, Detroit, MI, USA
Software change

- Change request
- Concept location
- Impact analysis
- Implementation
- Change Propagation
- Testing
IR-based concept location

Query

Ranked list of results

Source code

```
class ContextInformationPopup implements IContentAssistListener {
  public int fEngineOffset;
  public int fOffset;
  public int fVisibleOffset;
  public IContextInformation fInformations;
  public IContextInformationValidator fValidator;
  public IContextInformationPresenter fPresenter;
}
```
Challenge: the query

- Text in the query needs to match the text in the source code
- Difficult to formulate good queries
  - unfamiliar source code
  - unknown target
- hard to describe something that you do not know
Eclipse bug #13926

Bug description:

JFace Text Editor Leaves a Black Rectangle on Content Assist text insertion. Inserting a selected completion proposal from the context information popup causes a black rectangle to appear on top of the display.
Queries

• **Q1**: jface text editor black rectangle insert text

• **Q2**: jface text editor rectangle insert context information

• **Q3**: jface text editor content assist

• **Q4**: jface insert selected completion proposal context information
Queries and results

• Q1: jface text editor black rectangle insert text
  – position of modified method: 7496
• Q2: jface text editor rectangle insert context information
  – position of modified method: 258
• Q3: jface text editor content assist
  – position of modified method: 119
• Q4: jface insert selected completion proposal context information
  – position of modified method: 723

Whole change request: 54
IR CL in unfamiliar software

Developers:

• Rarely begin with a good query: hard to choose the right words

• Analyze very briefly list of results before reformulating query

• Even after reformulation, vague idea of what to look for -> queries not always better

• Can recognize whether the results retrieved are relevant or not to the problem
Questions

• Is there a way to make the query formulation easier on the developers?

• Is there a way to ensure that the subsequent queries lead us in the right direction?

• Can we do this by following the common practices of the developers?

• Can we improve IR-based CL using this approach?
Relevance feedback

- Uses developer feedback about relevancy of returned results to automatically reformulate queries
- Queries are reformulated by:
  - Adding terms from relevant documents
  - Removing terms from irrelevant documents
- Iterative process
- Common technique in text retrieval
- Used also in SE
JFace Text Editor Leaves a Black Rectangle on Content Assist text insertion. Inserting a selected completion proposal from the context information popup causes a black rectangle to appear on top of the display.

1. `createContextInfoPopup()` in `org.eclipse.jface.text.contentassist.ContextInformationPopup`
2. `configure()` in `org.eclipse.jdt.internal.debug.ui.JDIContentAssistPreference`
3. `showContextProposals()` in `org.eclipse.jface.text.contentassist.ContentInformationPopup`

New Query
IRRF tool

• IR Engine: *Lucene*
  – based on the Vector Space Model (VSM)
  – input: *methods, query*
  – output: a ranked list of methods ordered by their textual similarity to the query

• Relevance feedback: *Rocchio algorithm*
  – the classic algorithm for RF; used also in SE
  – models a way of incorporating relevance feedback information into the VSM
Evaluation

• Extracted bug descriptions and set of methods modified in the bug fixes from bug tracking systems
• Consider bug descriptions as initial queries for IR
• Measure #methods investigated until reaching a modified method before and after using RF
• Relevance feedback:
  – one developer provides feedback
  – feedback round ends after marking N methods as relevant or irrelevant (N = 1, 3, 5)
Stop criteria

• Target method in top N results

• More than 50 methods analyzed

• Position of target methods in the ranked list of results increases for 2 consecutive rounds -> query moving away from wanted methods
# Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Vers.</th>
<th>LOC</th>
<th>Methods</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>2.0</td>
<td>2,500,000</td>
<td>74,996</td>
<td>7,500</td>
</tr>
<tr>
<td>jEdit</td>
<td>4.2</td>
<td>300,000</td>
<td>5,366</td>
<td>750</td>
</tr>
<tr>
<td>Adempiere</td>
<td>3.1.0</td>
<td>330,000</td>
<td>28,622</td>
<td>1,900</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>System</th>
<th>RF improves IR</th>
<th>RF does not improve IR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>jEdit</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Adempiere</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>All</td>
<td>13</td>
<td>5</td>
</tr>
</tbody>
</table>
# Results

- **Eclipse:**

<table>
<thead>
<tr>
<th>Report #</th>
<th>Baseline</th>
<th>IRRF N=1</th>
<th>IRRF N=3</th>
<th>IRRF N=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>19686</td>
<td>428</td>
<td>453 (5r)</td>
<td>48 (16r)</td>
<td>46 m(9r)</td>
</tr>
</tbody>
</table>

- **jEdit:**

<table>
<thead>
<tr>
<th>Report #</th>
<th>Baseline</th>
<th>IRRF N=1</th>
<th>IRRF N=3</th>
<th>IRRF N=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1607211</td>
<td>354</td>
<td>103(5r)</td>
<td>36 (12r)</td>
<td>28 (6r)</td>
</tr>
</tbody>
</table>

- **Adempiere:**

<table>
<thead>
<tr>
<th>Report #</th>
<th>Baseline</th>
<th>IRRF N=1</th>
<th>IRRF N=3</th>
<th>IRRF N=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1628050</td>
<td>52</td>
<td>3 (3r)</td>
<td>5 (2r)</td>
<td>7 (2r)</td>
</tr>
</tbody>
</table>
Questions – revisited (1)

- Is there a way to make the query formulation easier on the developers?
  - automatic query formulation

- Is there a way to ensure that the subsequent queries lead us in the right direction?
  - add terms from relevant documents, remove terms from irrelevant documents
  - stop when we move away from the target (results worsen for 2 consecutive rounds)
Questions – revisited (2)

• Can we do this by following the common practices of the developers?
  – developers still analyze only a few results in the result list before reformulation

• Can we improve IR-based CL using relevance feedback?
  – in some cases yes
Current and future work

- Studies involving more systems and more developers

- Automatically calibrating the parameters for a specific system and a specific set of change requests

- Study the circumstances when RF does not improve IR