An Information Retrieval Approach For Automatically Constructing Software Libraries

Y. Maarek, D. Berry, and G. Kaiser

James

Department of Computer Sci&Eng
Mississippi State University
An adequate library for reuse purpose:

1. Provides a sufficient number of components
2. Well organized such that the existing code closest to the users’ needs is easy to locate.

• Goal of this paper: focus on the second issue with representing, storing, and retrieving reusable components.
# Approaches for building reuse systems

1. **Information retrieval (IR): the free-text indexing**

2. **Artificial intelligence (AI): knowledge-based**

<table>
<thead>
<tr>
<th></th>
<th>AI</th>
<th>IR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic?</strong></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Aim?</strong></td>
<td>Understand the queries</td>
<td>Characterize the doc.</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td><strong>Transportable?</strong></td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Scalable?</strong></td>
<td>Not good</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Lexical Affinity: LA

- A lexical relation between two units of language stands for a correlation of their common appearance in the utterances of the language.
- By Martin et al., 98% of lexical relations relate words which are separated by at most five words within a single sentence.
Calculations and formulas

- \[ \text{INFO}(w) = -\log_2(P\{w\}) \]
- \[ \text{INFO}(\{w_1, w_2\}) = -\log_2(P\{w_1, w_2\}) \]
- \[ \text{INFO}(\{w_1, w_2\}) = -\log_2(P\{w_1\} \times P\{w_2\}) \]
- \[ \rho(\{w_1, w_2, f\}) = f \times \text{INFO}(\{w_1, w_2\}) \]
- \[ \rho_z = (\rho - \bar{\rho})/\sigma \]

Q: Please explain what exactly the normalized resolving power is doing?
Q: Explain the figures 4 and 5 of the paper
A scenario for retrieving a component

1) **Query specification**: The user expresses a query according to the authorized vocabulary/nature language

2) **Linear retrieval**: A search locates the candidate components and the candidates are ranked according to their degree of match with the query

3) **Browsing**: Cluster-based retrieval is initiated when no adequate components have been found by the linear retrieval.
Empirical results

- User effort
- Maintenance effort
- Efficiency
- Retrieval effectiveness:
  \[ \text{recall} = \frac{r}{R} \]
  \[ \text{precision} = \frac{r}{C} \]
Questions:

1. After linear based retrieval, how it determined up to which hierarchical level the cluster based retrieval should be considered?

2. How do you find the measure of similarities in the IR method of the linear retrieval step?

3. A high-level languages have an advantage over low-level languages as far as searchability goes?

4. Are there any other kinds of indexing algorithms that are more efficient than this?

5. What’s the significance of using Natural-language documentation?

6. Since this paper was published in 1991, how effective are these ideas today?

7. The author mentions on page 6, under the heading 'The Classifying Stage' that a component must match the query exactly for it to be retrieved. I know this is true in some cases, but aren't there methods of searching through a database that will retrieve close matches? If so, could these retrieval methods be applied to a database?

8. How are profiles stored for each file? Are these text files? If one profile is required for each file—does this demand extra system resources?

9. How could this method be used for poor programmers who do not comment their code sufficiently?

10. Can you please discuss about the term “phrase generation” approach used in Indexing (section III –A)?

is based on a simple language analysis system that determines the syntactic function of individual text words with a high degree of accuracy, and chooses of indexing phrases based on weights assigned to the phrase components. The Until sophisticated semantic components can be used as part of a language analysis system, it is safer to remain with the conceptionally simpler approaches that tend to be more forgiving for general-purpose texts.
Questions continue...

- An example of the sliding window technique
- Can a similar approach be adopted to retrieve requirements instead of software components?
- Do you think limiting the user freedom in forming the query will affect the efficiency?
- Do you think the introduction of facetted classification techniques will increase the efficiency?
Thank you