Clustering to support program comprehension
Outline

• What is program comprehension?
• Applications of clustering.
• Issues and challenges.
• Future work.
We spent half the time understanding a program

Why?

Documentation is out of date.

Software continuously evolves.

Nobody cares about updating the documentation (it is boring).

Documentation keeps outdated.
PC approaches help to retrieve knowledge from software artifacts (mostly source code)

When?

New developers in the team.

We always forget our previous code!!

Refactoring.

We need fast (aka automatic) techniques to retrieve implicit knowledge from source code
Architecture = High-level overview = Clustering

Decompose a system into meaningful subsystems

Automatically identify groups of related/similar source code entities

CLUSTERING!!!
Similarity between SC entities

Structural information (Relationships among entities)

Class A1 → Class A2

Class B1 → Class B2 → Class B3 → Class C1

Semantic information (Similarity in the vocabulary)

Query
- Class A1
- Class A2

DB Conn
- Class A1
- Class A2

Logging
- Class C1

FileManager
- Class B1
- Class B2

Serializer
- Class B1
- Class B2

Stream
## Similarity using structural information

### Binary/categorical relationships between classes

<table>
<thead>
<tr>
<th></th>
<th>Class B1</th>
<th>Class B2</th>
<th>Class B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Class B2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Class B3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1 = Class $i$ imports/uses/calls Class $j$

### Similarity measure

<table>
<thead>
<tr>
<th>Similarity measure</th>
<th>Formula for $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple matching coefficient</td>
<td>$\frac{a + d}{a + b + c + d}$</td>
</tr>
<tr>
<td>Jaccard coefficient</td>
<td>$\frac{a}{a + b + c}$</td>
</tr>
<tr>
<td>Sorensen coefficient</td>
<td>$\frac{2a}{2a + b + c}$</td>
</tr>
</tbody>
</table>

- $a$ = # of attributes with 1 in both classes
- $b, c$ = # of attributes with 1 in one class and 0 in the other
- $d$ = # of attributes with 0 in both classes
Similarity using structural information

Numerical relationships between classes

<table>
<thead>
<tr>
<th></th>
<th>Class B1</th>
<th>Class B2</th>
<th>Class B3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class B1</td>
<td>$\omega_{11}$</td>
<td>$\omega_{12}$</td>
<td>$\omega_{13}$</td>
</tr>
<tr>
<td>Class B2</td>
<td>$\omega_{21}$</td>
<td>$\omega_{22}$</td>
<td>$\omega_{23}$</td>
</tr>
<tr>
<td>Class B3</td>
<td>$\omega_{31}$</td>
<td>$\omega_{32}$</td>
<td>$\omega_{33}$</td>
</tr>
</tbody>
</table>

Similarity measure

<table>
<thead>
<tr>
<th></th>
<th>Formula for $d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$</td>
</tr>
<tr>
<td>Maximum</td>
<td>$\max</td>
</tr>
<tr>
<td>Manhattan</td>
<td>$\sum_{i=1}^{n}</td>
</tr>
</tbody>
</table>
Similarity using semantic information

- We use tools designed for information retrieval and NLP
- Source code file = text document
- We are interested in the number of occurrences of each word inside a document.

Which class is more similar to “New Class”?
Similarity using semantic information

- We use tools designed for information retrieval and NLP
- Source code file = text document
- We are interested in the number of occurrences of each word inside a document.

**Cosine similarity:**

\[ \frac{A \cdot B}{||A|| \cdot ||B||} \]

In general, the frequency of a word is not a good criterion to compute similarity. So, other relevance measures are used (e.g. TF/IDF).
With a similarity matrix you can use clustering

- **Complete linkage as hierarchical algorithm.** [Anquetil and Lethbridge 1999]
- **Single Linkage as hierarchical algorithm.** [Anquetil and Lethbridge 1999]
- **Weighted Combined Algorithm (Linkage based)** [Maqbool and Babri 2004]
- **K-Medoids** [Corazza et al 2010]
- **Hierarchical Clustering** [Garcia et al 2011, Corazza et al 2011]
We do not know what is the best algorithm…

• Researchers have chosen clustering algorithms based on its performance or reputation.

  [Corrazza 2011] changed from k-medoids to HAC arguing that the latest is deterministic and faster.

  [Kuhn 2007] chose AL because it is an “standard algorithm”.

• We “forgot” that different algorithms can provide different clusterings from the same dataset.
... so we must trust in “quality measures”...

- Human judgment.
  - It is difficult to visualize the results (non-Euclidean spaces, high dimensional data).

- Heuristics
  - Arbitrary cutpoint in a dendogram.
  - Good clustering: $|C| \approx |X| \times 0.1$

- Clustering quality measures.
  - Some times to choose $k$ (e.g. elbow method, silhouette coefficient).
... and we mostly trust in our own “quality measures”.

- **Authoritativeness**: Clustering distance between the output of an algorithm and a ground truth.
  - The ground truth is really hard to build.
  - Software changes, so the ground truth becomes outdated.

- **Non-Extremity**: The number of clusters must be meaningful (no big or small clusters).

- **Stability**: Small changes in the dataset must not perturb the clustering.
  - Again, software continuously changes.
Conclusion: We need to choose wisely.

- Recent work have focused on improving authoritativeness.
  - Different developers can provide different ground truths.
  - Authoritativeness depends on the clusterability of the data.

- We need to know if our similarity measures can appropriately capture the structure of the data.

- With a good similarity measure, we can proceed to choose the best clustering algorithm.