Enhancing Computer Science Education by Automated Analysis of Students' Code Submissions

ECAI 2023 - Workshop AI4AI

Lea Eileen Brauner & Frank Höppner

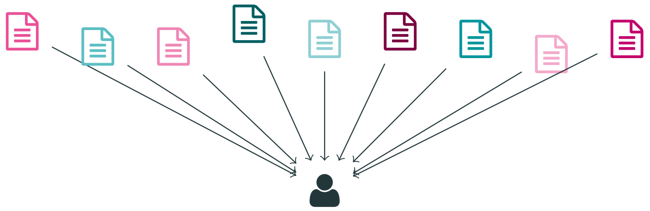
Ostfalia University of Applied Sciences
Dept. of Computer Science, Wolfenbüttel, Germany

Agenda

1. Motivation & Problem
2. Challenges & Related Areas
3. Proposed Approach
4. Evaluation
5. Summary

Motivation & Problem

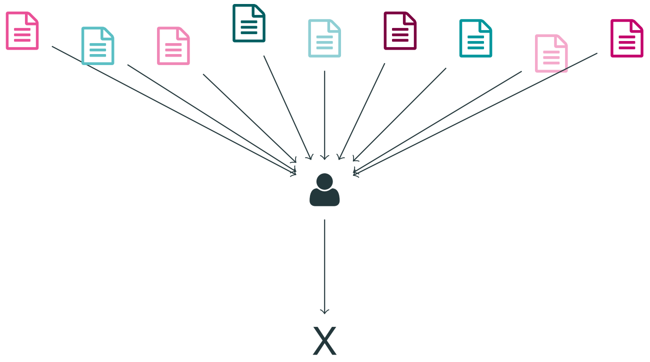
Motivation



students

lecturer

Motivation

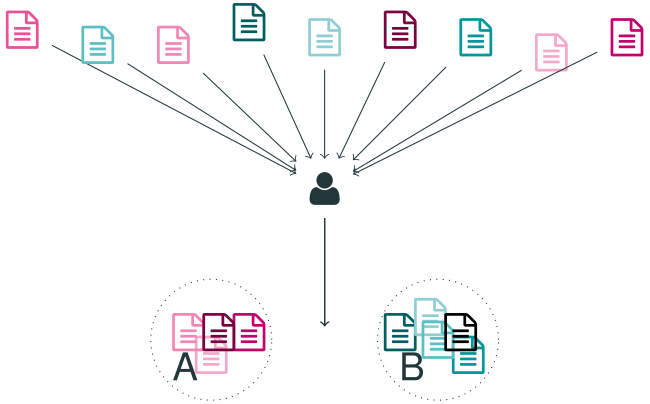


students

lecturer

grading? unit-testing?

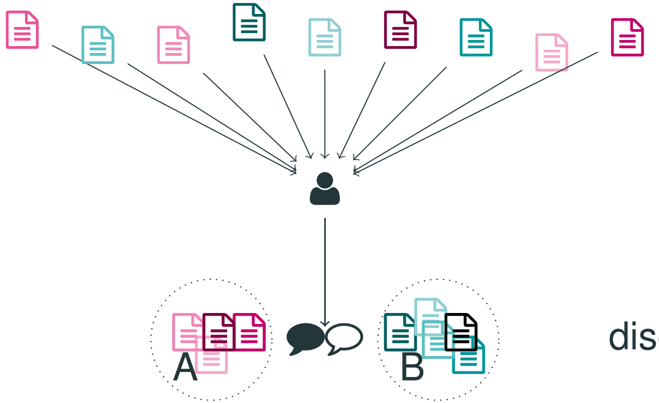
Motivation



students

lecturer

Motivation

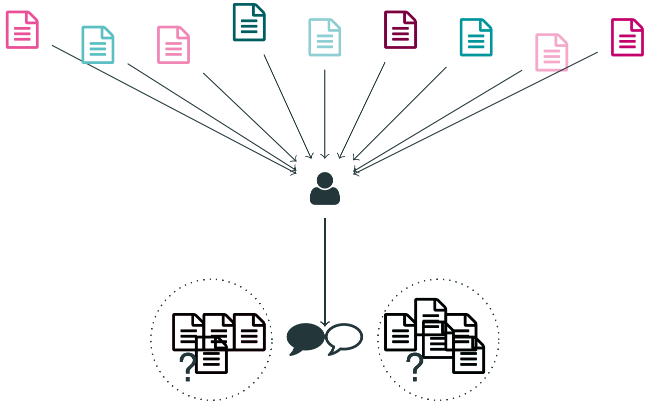


students

lecturer

discuss solution approaches!

Problem



students

lecturer

black boxes

Challenges & Related Areas

Challenges & Related Areas

task: write a function that returns the value range of the passed array (which carries elements of up to three digits only)

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

Challenges & Related Areas

task: write a function that returns the value range of the passed array (which carries elements of up to three digits only)

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

different approaches:

- sort array first, subtract first from last element
- first determine only the maximum, then negate array, again determine only the maximum (now minimum)

Challenges & Related Areas

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

```
class Range {
  int small = 1000;
  int large = -1000;

  void include(int a) {
    if (a<small) small=a;
    else if (a>large) large=a;
  }
  int range(int[] arr) {
    for (int i=0;i<arr.length;++i)
      include(arr[i]);
    return large-small;
} }
```

Challenges & Related Areas

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

```
class Range {
  int small = 1000;
  int large = -1000;

  void include(int a) {
    if (a<small) small=a;
    else if (a>large) large=a;
  }
  int range(int[] arr) {
    for (int i=0;i<arr.length;++i)
      include(arr[i]);
    return large-small;
} }
```

Related Areas

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

- plagiarism detection
- edit distance
- code clone detection

```
class Range {
  int small = 1000;
  int large = -1000;

  void include(int a) {
    if (a<small) small=a;
    else if (a>large) large=a;
  }
  int range(int[] arr) {
    for (int i=0;i<arr.length;++i)
      include(arr[i]);
    return large-small;
} }
```

Related Areas

```
class Range {

  int getRange(int[] arr) {
    int min = 1000;
    int max = -1000;
    for (int i=0;i<arr.length;++i)
      if (arr[i]<min) min=arr[i];
      else if (arr[i]>max) max=arr[i];
    return max-min;
  }
}
```

```
class Range {
  int small = 1000;
  int large = -1000;

  void include(int a) {
    if (a<small) small=a;
    else if (a>large) large=a;
  }
  int range(int[] arr) {
    for (int i=0;i<arr.length;++i)
      include(arr[i]);
    return large-small;
} }
```

- plagiarism detection ✗
- edit distance ✗
- code clone detection ✗
- idea: involve variable usage

Proposed Approach

Variable Usage Paths (VUPs)

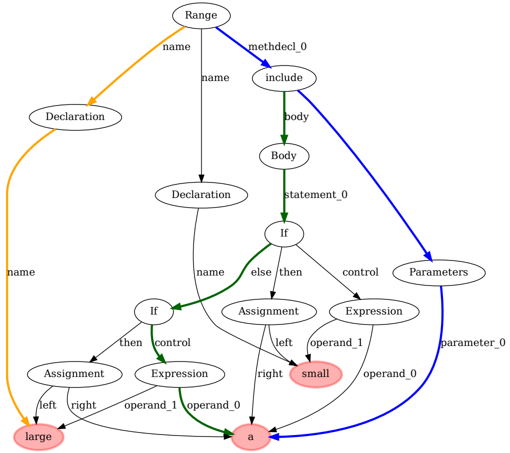


Figure 1: modified AST of class Range

```
class Range {
  int small = 1000;
  int large = -1000;

  void include(int a) {
    if (a<small) small=a;
    else if (a>large) large=a;
  }...
}
```

→ a/parameters/include/Range

→ a/expression/if/if/body/include/Range

→ large/declaration/Range

Class Comparison

class C	class D
a/expression/while/func/C	x/assignment/for/proc/D
a/declaration/func/C	x/expression/for/proc/D
b/assignment/for/func/C	x/declaration/D
b/expression/for/func/C	y/expression/for/read/D
b/declaration/func/C	y/declaration/read/D
c/expression/for/get/C	z/expression/while/proc/D
c/declaration/get/C	z/declaration/proc/D

Class Comparison

class C

a/expression/while/func/C
a/declaration/func/C

b/assignment/for/func/C
b/expression/for/func/C
b/declaration/func/C

c/expression/for/get/C
c/declaration/get/C

class D

x/assignment/for/proc/D
x/expression/for/proc/D
x/declaration/D

y/expression/for/read/D
y/declaration/read/D

z/expression/while/proc/D
z/declaration/proc/D

Class Comparison

class C

a/expression/while/func/C
a/declaration/func/C

b/assignment/for/func/C
b/expression/for/func/C
b/declaration/func/C

c/expression/for/get/C
c/declaration/get/C

sim=$\frac{2}{3}$

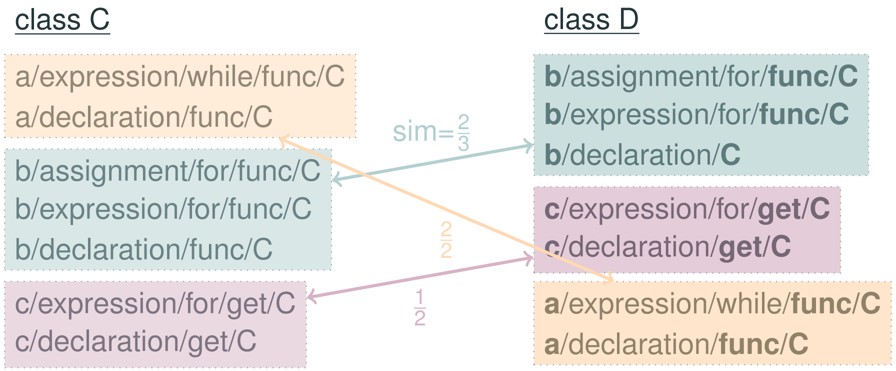
class D

b/assignment/for/**func/C**
b/expression/for/**func/C**
b/declaration/D

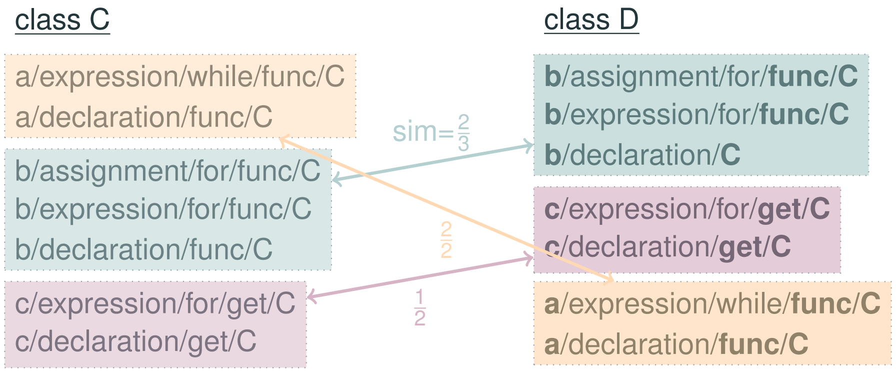
y/expression/for/read/D
y/declaration/read/D

z/expression/while/proc/D
z/declaration/proc/D

Class Comparison


class C
a/expression/while/func/C
a/declaration/func/C
b/assignment/for/func/C
b/expression/for/func/C
b/declaration/func/C
c/expression/for/get/C
c/declaration/get/C
class D
b/assignment/for/**func**/**C**
b/expression/for/**func**/**C**
b/declaration/**C**
c/expression/for/**get**/**C**
c/declaration/**get**/**C**
a/expression/while/**func**/**C**
a/declaration/**func**/**C**
sim=$\frac{2}{3}$
$\frac{2}{2}$
$\frac{1}{2}$

Class Comparison



total sim = $\frac{5}{7}$

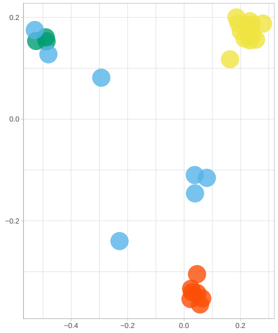
Evaluation

Evaluation - Procedure

- 2-step-Evaluation
  1. self-created example programmes $\rightarrow$ explicit testing of desired behaviour
  2. more comprehensive evaluation on real student submissions & comparison of different approaches and JPlag

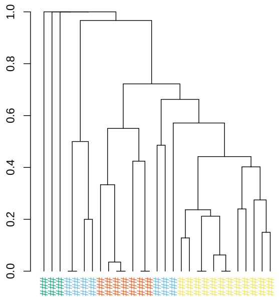
Evaluation - Procedure

- 2-step-Evaluation
- projection of results in 2D-space

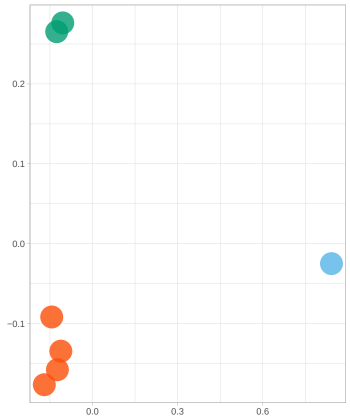


Evaluation - Procedure

- 2-step-Evaluation
- projection of results in 2D-space
- manual grouping in comparison with hierachical cluster analysis



Evaluation - Results



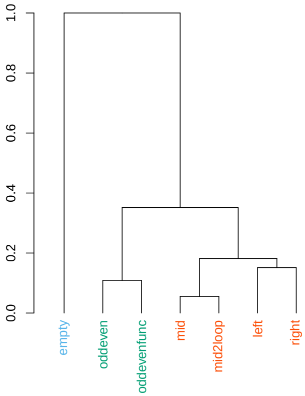
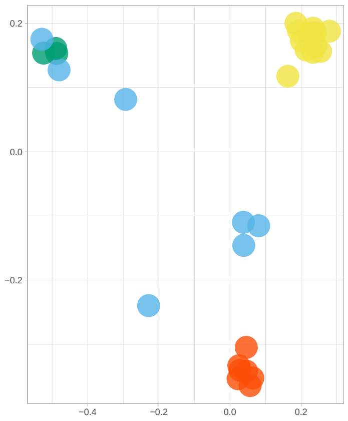


Figure 2: Evaluation of example codes

Evaluation - Results



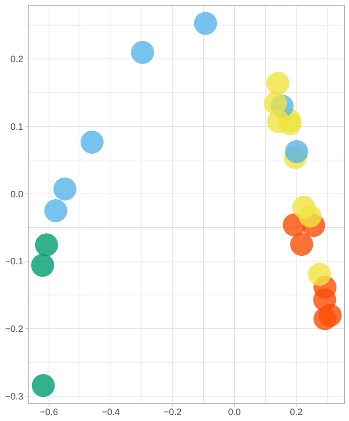
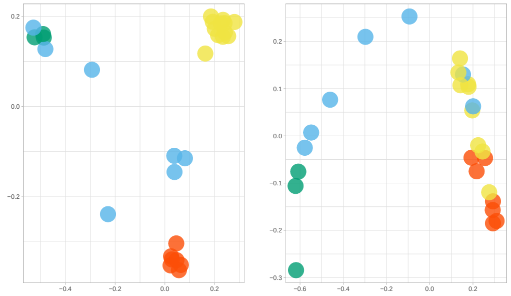
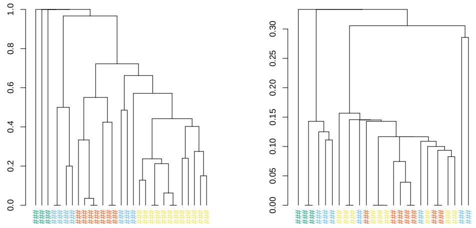


Figure 3: 2D results of proposed approach (left), Jplag (right)

Evaluation - Results

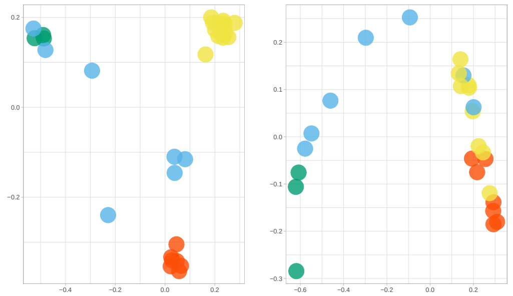


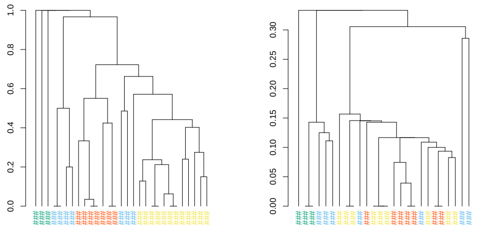
(a) 2D results of proposed approach (left), Jplag (right)



(b) dendrograms of hierachical clustering of proposed approach (left), Jplag (right)

Evaluation - Results





(a) 2D results of proposed approach (left), Jplag (right)

(b) dendrograms of hierachical clustering of proposed approach (left), Jplag (right)

→ solution approaches can be distinguished by comparing VUPs

Summary

Summary

Summary:

- Measure structural similarity of student submissions
- Classical approaches fail (edit distance, JPlag)
- New approach guided by variable usage
- Encouraging results

Thank you for your attention!

Any questions?