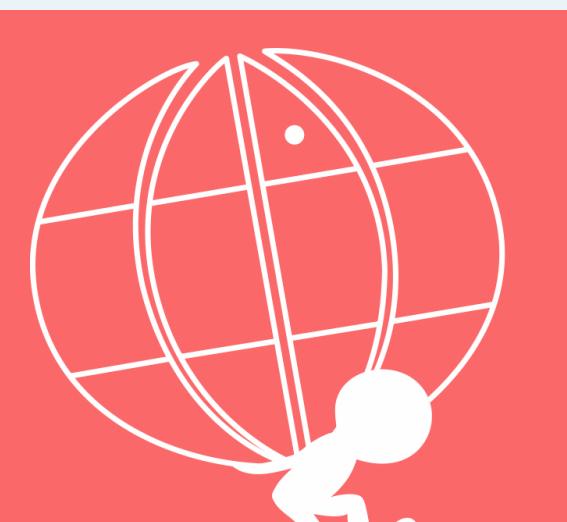


# Usability of Anchoring Algorithms for Source Code

Karol Rástočný  
Supervisor: Prof. Mária Bieliková



PeWe@FIIT  
personalized web group

## Problem

- Anchors useable for machine learning without source code
- Accurate method for anchoring in source code with real-time usability

## Location Descriptor

- Index-based location descriptor
  - indexes of the first and the last letter
- Context-based location descriptor
  - Tagged text
    - sequence of tagged textual elements
  - Context before tagged text
    - minimal unique sequence directly before tagged text
  - Context after tagged text
    - minimal unique sequence directly before tagged text

## Algorithm

1. Pre-process a source code
  - the source code is spitted to textual element
2. Compare textual elements
  - unique textual elements are compared by a string similarity algorithm (letter to letter comparison)
3. Local sequence alignment
  - locations of a tagged text are searched as sub-sequences of the source code via the Smith-Waterman algorithm (element to element comparison)
4. Calculate scores of matches for contexts
  - scores of contexts before and after tagged text are calculated for each possible alignment of tagged text
5. Calculate confidences of matched locations

$$\text{Confidence} = \frac{sc_{\text{context before}} + sc_{\text{tagged text}} + sc_{\text{context after}}}{5}$$

## Solution

- Context-based location descriptors
- Break up approximate string matching to two smaller parts

## Evaluation

- Goals
  - Acknowledgement of usability for real-time processing
  - Acknowledgement of usability for accurate precision in source code
  - Measure precision and time and memory complexity
- Test cases
  - TC I: Add line before tagged position
  - TC II: Add line before and after tagged position
  - TC III: Add three lines before tagged position
  - TC IV: Add three lines before and after tagged position
  - TC V: Delete line before tagged position
  - TC VI: Delete line before and after tagged position
  - TC VII: Delete three lines before tagged position
  - TC VIII: Delete three lines before and after tagged position
  - TC IX: Misspell correction
  - TC X: Rename an attribute/method
  - TC XI: Copy the tagged text four lines before the original position
  - TC XII: Modify a half of the tagged text

## Results

- The best speed
  - Textual element: Line
  - Similarity algorithm: Jaro-Winkler
- Acceptable speed with acceptable accuracy
  - Textual element: Word
  - Similarity algorithm: Jaro-Winkler
- The best accuracy
  - Textual element: Word
  - Similarity algorithm: Monge-Elkan

## Original source code

```
public double Stats(int[] numbers, out int min, out int max)
{
    double sum = 0;
    min = int.MaxValue;
    max = int.MinValue;
    foreach (var nber in numbers)
        min = Math.Min(min, nber);
    foreach (int nber in numbers)
        sum += nber;
    foreach (var nber in numbers)
        max = Math.Max(max, nber);
    return sum / numbers.Length;
}
```

Copied from:  
www.codeplex.com...

## Modified source code

```
public double Stats(int[] numbers, out int min, out int max)
{
    double sum = 0;
    min = int.MaxValue;
    max = int.MinValue;
    ①b foreach (var number in numbers)
        ①t min = Math.Min(min, number);
    ①a foreach (int number in numbers)
        sum += number;
    ②b foreach (var number in numbers)
        ②t max = Math.Max(max, number);
    ②a return sum / numbers.Length;
}
```

## Location descriptor

Indexes	[173; 199]
Context before	MinValue ; foreach ( var nber in numbers )
Tagged text	min = Math . Min ( min , nber ) ;
Context after	foreach ( int

## 1. Pre-process a source code

```
public double stats ( int [ ] numbers , out int min , out int max )
double sum = 0 ; min = int . MaxValue ; max = int . MinValue ; ...
```

## 2. Compare textual elements

	numbers	number	return	Max	Min	min	max	
nber	0.5714	0.6667	0.1667	Min	0.33	1	0.67	0
foreach	0	0	0.1429	Max	1	0.33	0	0.67

## 3. Local sequence alignment

2t	max	=	Math	.	Max	(	max	,	number	)	;	8.67
tt	min	=	Math	.	Min	(	min	,	nber	)	;	11
Sc.	0.33	1.33	2.33	3.33	3.67	4.67	5	6	6.67	7.67	8.67	0.788

## 4. Calculate scores of matches for contexts

2b	)	numbers	in	nber	var	foreach	;	numbers	...	7
cb	)	numbers	in	nber	var	foreach	;	MinValue	MinValue	8
Sc.	1	2	3	4	5	6	7	6	5, 4, ..., 0	0.875

## 5. Calculate confidences of matched locations

	b	t	a	Score
1	0.958	0.97	1	0.974
2	0.875	0.788	0.048	0.657