

Complexity-Based Code Embeddings

Rares Folea, Radu Iacob, Emil Slusanschi, Traian Rebedea

Faculty of Automatic Control and Computers
University Politehnica of Bucharest

Research objective:

**Can we provide a generic way of
converting **code** into **numerical
embeddings** meaningful for generic
analysis?**

Evaluation task:

Can we **estimate** what class of problems does a **code** solve?

Mathematical problems

```
if (i % 2 == 0) {  
    return 4  
} else { return 6 }
```

Dynamic programming

```
d[n]=d[n-1]+2*d[n-2]
```

Graph theory

```
for(k=1; k<n; k++) {  
    a[i][j] = a[i][k] +  
    a[k][j]  
}
```

Brute force

```
for(i=1;i<n;i++){  
    for(j=1;j<n;j++){  
        for(k=1;k<n;k++){  
            ...}}}
```

Word embeddings: mapping between words and numerical vectors

`"mother" = [0.213, 0.002, 0.889, 0.110, 0.553, ..., 0.941]`

`"father" = [0.331, 0.122, 0.189, 0.117, 0.923, ..., 0.822]`

Core idea:

Create code **embeddings** using **dynamic** profiling over code snippets.

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
int main() {
```

```
    int m, n;
```

```
    cin >> n >> m;
```

```
    int a;
```

```
    cin >> a;
```

```
    int k = ((n + (a - 1)) / a)  
            * ((m + (a - 1)) / a);
```

```
    cout << k;
```

```
}
```



| | | | | | |
|-------|-------|-------|-------|-------|-------|
| 0.631 | 0.431 | 0.221 | 0.332 | 0.209 | 0.992 |
|-------|-------|-------|-------|-------|-------|

Our solution for creating **embeddings**: (1)

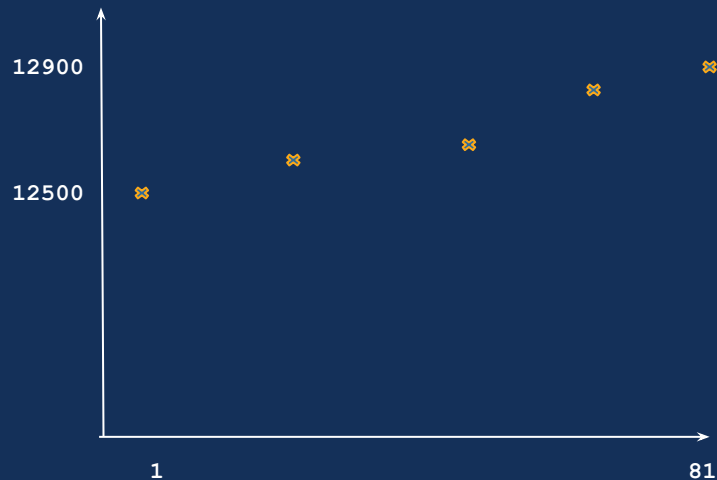
"branch-misses":

{ "1": 12500.0,

"21": 12605.2,

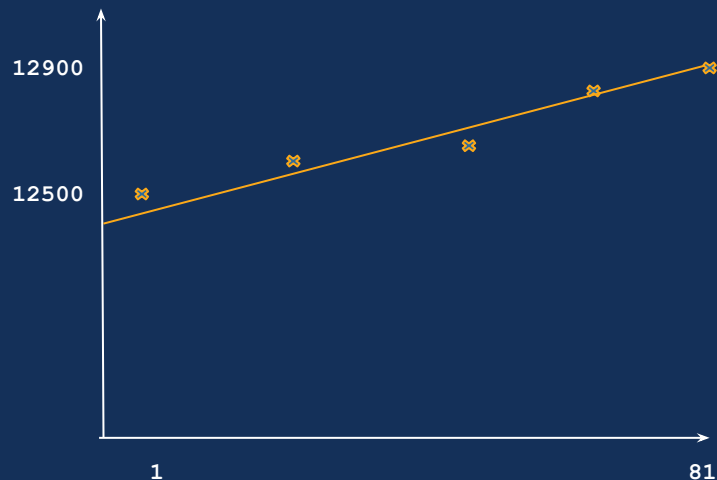
"41": 12706.1,

...



Our solution for creating **embeddings**: (2)

```
"branch-misses": {  
  "FEATURE_CONFIG": 1,  
  "FEATURE_TYPE": "POLYNOMIAL",  
  "INTERCEPT": 12475.13,  
  "R-VAL": 50.56  
},
```



Analysed metrics for building the code embeddings

- **branch-misses**
- **branches**
- **context-switches**
- **cycles**
- **instructions**
- **page-faults**

and more

**With the dynamic code
embeddings computed, we can
design classification models.**

Binary classification

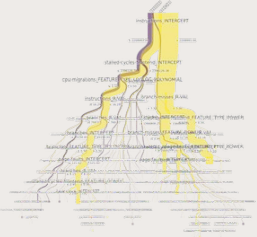
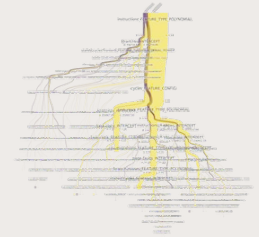
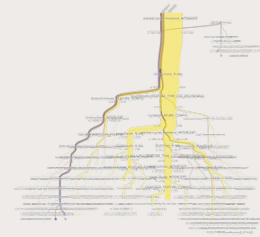
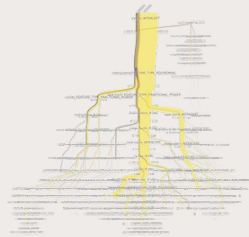
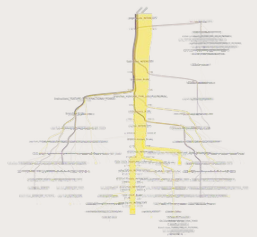
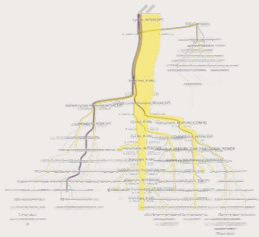
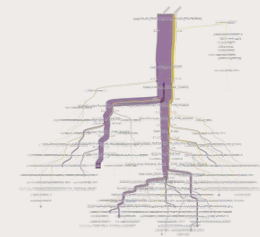
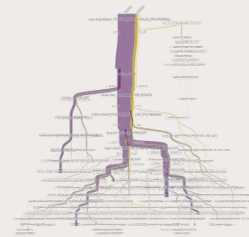
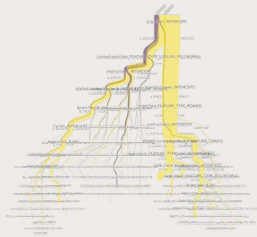
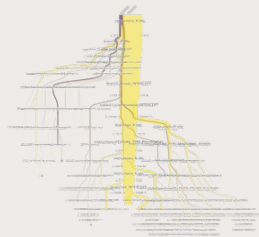
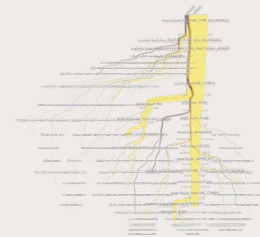
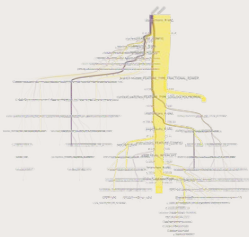
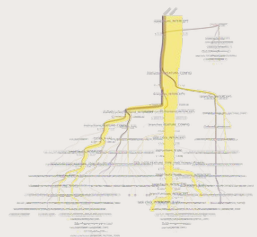
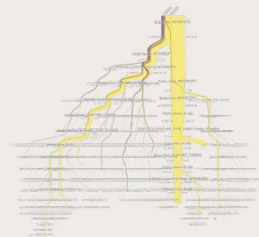
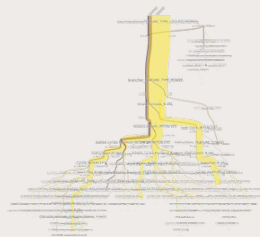
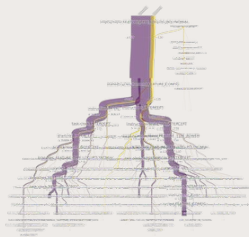
Decisions Tree

- **binary math/non-math classification**
- **96%+ accuracy on a testing dataset based on 5000+ open-source solutions from Codeforces**



Random forest

- **binary**
math/non-math
classification
- **between 16 and**
10000 classifiers
- **97%+ accuracy on**
testing dataset



Multi-label classification

XGBoost

- Achieving Multi Label Classification by training one classifier per target.
- Best performance, in terms of precision, recall and F1-score for the analyzed scenarios.

| Class | Precision | Recall | F-score | Support |
|--------------------|-------------|-------------|-------------|---------|
| strings | 0.94 | 0.9 | 0.92 | 756 |
| implementation | 0.94 | 0.98 | 0.96 | 1387 |
| greedy | 0.92 | 0.77 | 0.84 | 523 |
| brute force | 0.98 | 0.77 | 0.86 | 311 |
| dp | 0.87 | 0.74 | 0.8 | 35 |
| divide and conquer | 1.0 | 0.68 | 0.81 | 31 |
| graphs | 0.91 | 0.88 | 0.9 | 83 |
| binary search | 1.0 | 0.68 | 0.81 | 31 |
| math | 0.97 | 0.91 | 0.94 | 301 |
| sortings | 0.95 | 0.61 | 0.74 | 176 |
| shortest paths | 0.91 | 0.88 | 0.9 | 83 |
| micro avg | 0.94 | 0.88 | 0.91 | 3717 |
| macro avg | 0.94 | 0.8 | 0.86 | 3717 |
| weighted avg | 0.94 | 0.88 | 0.91 | 3717 |
| avg | 0.94 | 0.91 | 0.91 | 3717 |

Future possible application:




How will the algorithm **behave** wrt. a given input dimension / architecture?



Malware detection: is running a snippet dangerous?



Better understanding in **what** a specific snippet does.



Code similarity metrics: How likely is this code plagiarized?

Thank you!

Follow **the models research** development:

<https://github.com/raresraf/AlgoRAF>

Follow **the embeddings research** development:

<https://github.com/raresraf/rafPipeline>

Contribute to our dataset:

<https://github.com/raresraf/TheInputsCodeforces>