

De-anonymizing Programmers via Code Stylometry

(USENIX 15')

Aylin Caliskan-Islam
Drexel University

Richard Harang
U.S. Army Research Laboratory

Andrew Liu
University of Maryland

Arvind Narayanan
Princeton University

Clare Voss
U.S. Army Research Laboratory

Fabian Yamaguchi
University of Goettingen

Rachel Greenstadt
Drexel University

Presenter: Dongok Kim

KAIST

#Hacking Lab

Introduction

Stylometry,
Anonymity and
De-anonymization

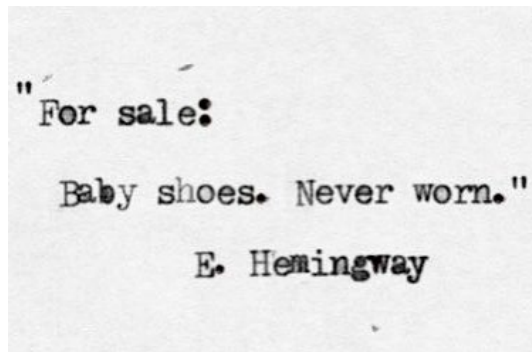
Stylometry

Stylometry

> The **statistical analysis of variations** in literary style between one writer or genre and another.



Art



Literature



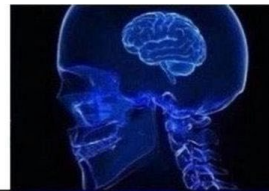
Chess play(?)

Source code Stylometry

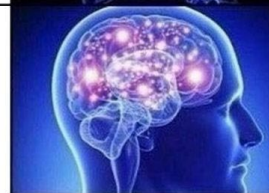
- Everyone has unique coding tastes
 - Indentation character, newline before brackets...
 - Monolithic/modular, for/while...

- Source code can also incorporate stylometry

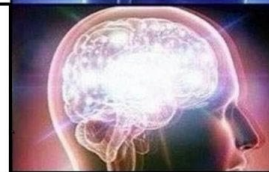
```
if (var) {  
    foo();  
} else {  
    bar();  
}
```



```
if (var)  
{  
    foo();  
}  
else  
{  
    bar();  
}
```



```
if  
(  
    var  
)  
{  
    foo();  
}  
else  
{  
    bar();  
}
```

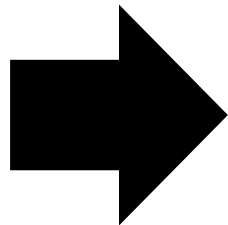


```
if  
{  
    var  
}  
foo  
{  
}  
}  
else  
{  
    bar  
{  
}  
}
```

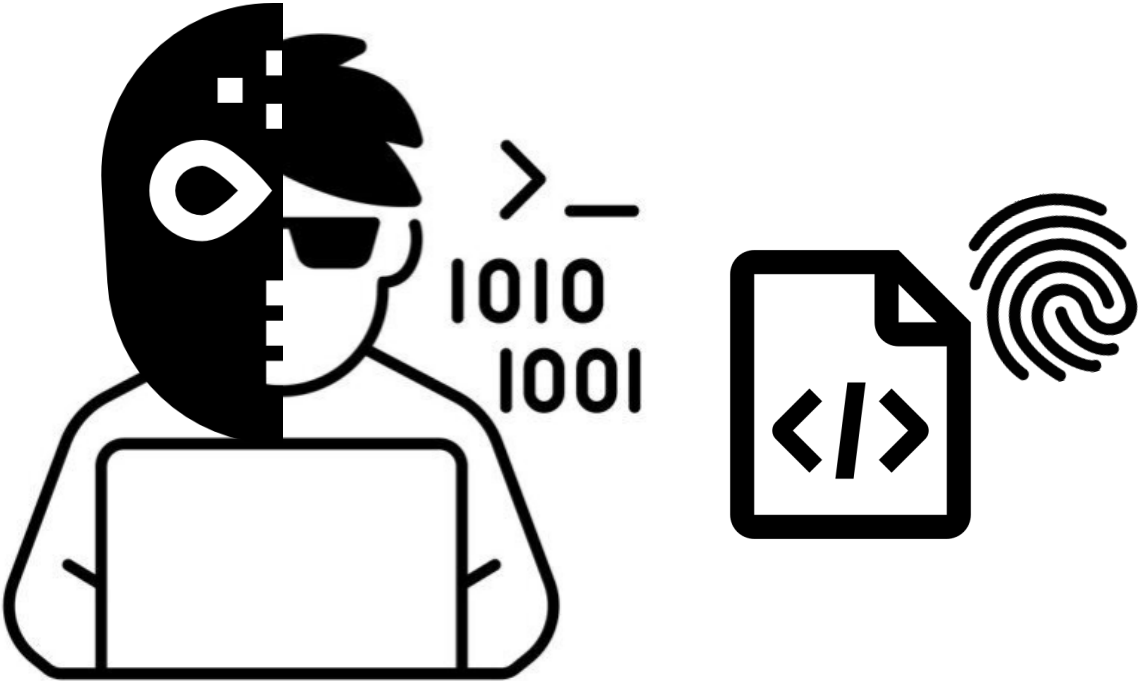


Anonymity and De-anonymization

- Anonymity == Indistinguishability
- De-anonymization
== Distinguish something from (should be) indistinguishable set



De-anonymizing Programmers via Code Stylometry



Motivation

Why and How

Source code Stylometry and

Programmer de-anonymization is

Important?

Importance of Source code Privacy & De-anonymization

Satoshi Nakamoto
Born: Unknown

Creator(s) of Bitcoin
Cryptocurrency

- Pseudonym; true identity has not been verified or revealed
- Authored the Bitcoin whitepaper
- Designed first blockchain database



Investopedia

VS



**WANTED
BY THE FBI**



KIM IL



PARK JIN HYOK

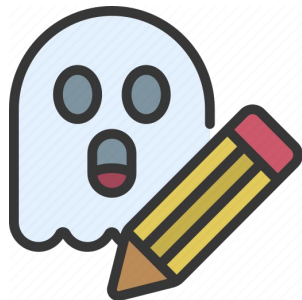


Source code stylometry
is the key of privacy & security

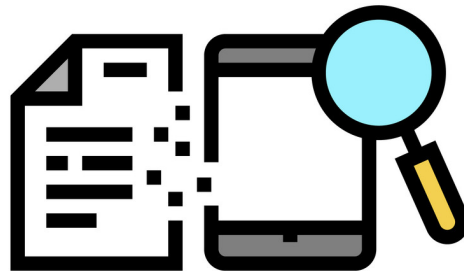
Analyst's interests



De-anonymization of
programmer



Ghostwriting detection



Software forensics



Copyright investigations

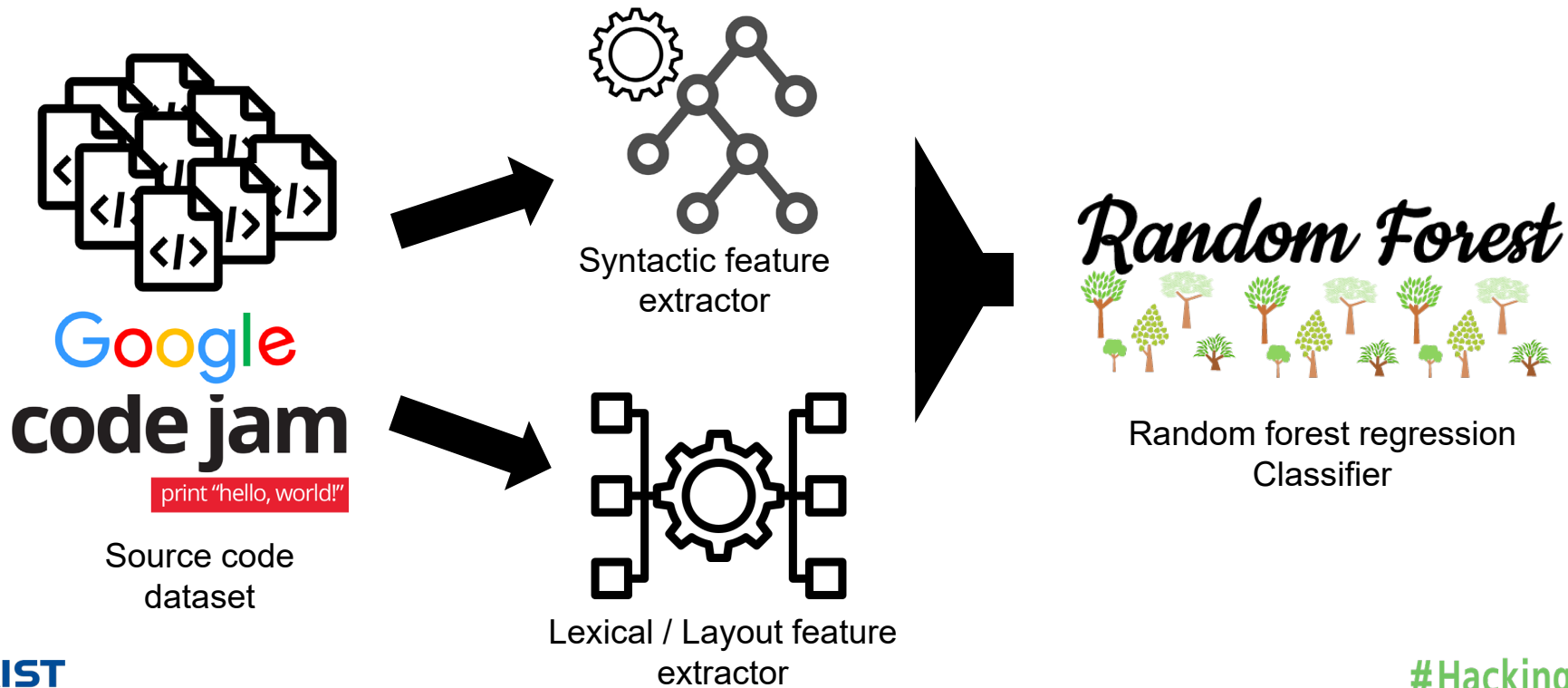


Authorship verification

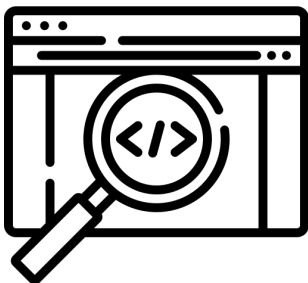
Approaches

SCFS and Random Forest
Approaches

Approaches Overview



Code Stylometry Feature Set (CSFS)



- Lexical feature

- # of ternary operations
- # of comments
- # of literals
- ...



- Layout feature

- # of tabs
- # of spaces
- # of empty line
- Bracelets before/after newline
- ...

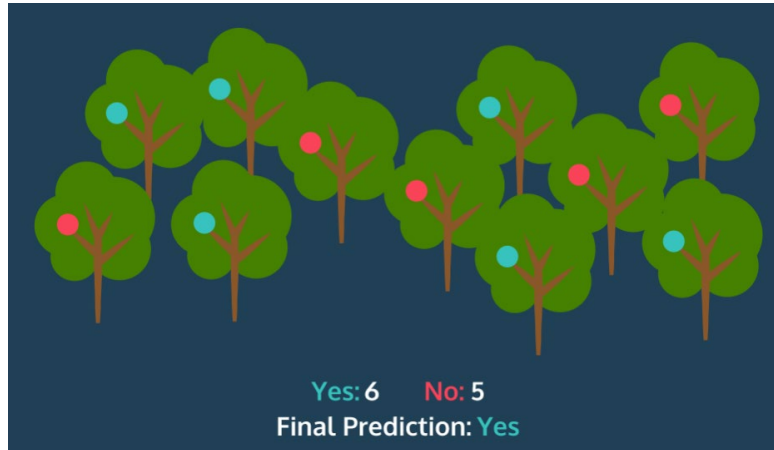


- Syntactic feature

- Max AST node depth
- Avg AST node depth
- C/C++ keyword
- ...

Random Forest Classification

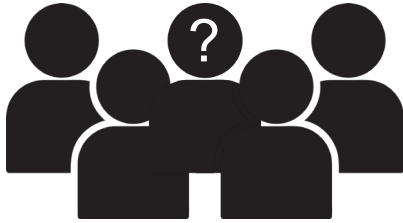
- Perform random forest classification
 - 300 decision trees
 - $\log(\# \text{ of total feature}) + 1$ random feature set



Evaluation

Effectiveness,
Robustness,
Generalizability and
Limitations

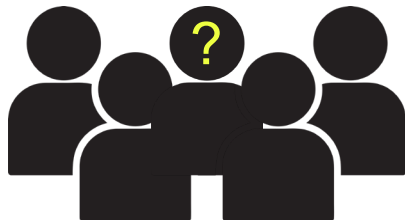
Effectiveness - Authorship Attribution



Multi-class
Closed World Task

A = #programmers, F = max #problems completed		
N = #problems included in dataset ($N \leq F$)		
A = 250 from 2014	A = 250 from 2012	A = 250 all years
F = 9 from 2014	F = 9 from 2014	F \geq 9 all years
N = 9	N = 9	N = 9
Average accuracy after 10 iterations with IG-CSFS features		
95.07%	96.83%	98.04%

Effectiveness - Authorship Attribution



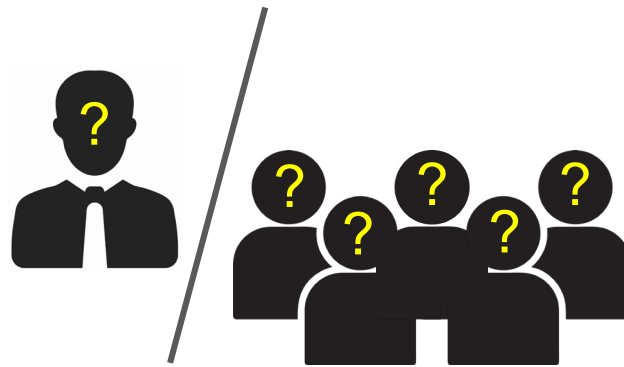
Multi-class
Open World Task

96.67%
on 30 classes



Two-class
Closed World Task

100%



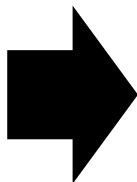
Two-class/One-class
Open World Task

100% for Mallory
82.04% for non-Mallory

Robustness - Obfuscation (STUNNIX)

```
#ifndef __STL_USE_EXCEPTIONS
extern void __out_of_range ( const char * );
#define OUTFRANGE( cond, msg) \
do { if ( cond) __out_of_range ( #cond); } while ( 0)
#else
#include <cassert>
#define OUTFRANGE( cond, msg) assert ( ! ( cond) )
#endif


template < class charT, class traits, class Allocator>
basic_string < charT, traits, Allocator> &
basic_string < charT, traits, Allocator> ::
replace ( size_type pos1, size_type n1,
```



```
#ifndef z7929401884
extern void za41dafc42e( const char* );
#define z1c52ffdd48( z22fc207d33, zde05b8b1b0) \
do { if ( z22fc207d33) za41dafc42e ( #z22fc207d33); } while ( (0x1fb1+1115-0x240c))
#else
#include <cassert>
#define z1c52ffdd48( z22fc207d33, zde05b8b1b0) z7bd0031cc2 ( ! ( z22fc207d33) )
#endif
template< class zd9cfc9cefe, class z9cdf2cd536, class Allocator> basic_string<
zd9cfc9cefe, z9cdf2cd536, Allocator> & basic_string< zd9cfc9cefe, z9cdf2cd536,
Allocator> :: replace ( size_type z795f772c7c, size_type zddd43c876a,
const basic_string& str, size_type z8ad17de27a, size_type za2e5f06cda) {
```

Obfuscator	Programmers	Lang	Results w/o Obfuscation	Results w/ Obfuscation
Stunnix	20	C++	98.89%	100.00%
Stunnix	20	C++	98.89*%	98.89*%

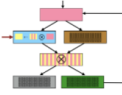
Robustness - Obfuscation (TIGRESS)



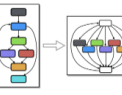
Virtualize
Turn a function into a specialized interpreter.



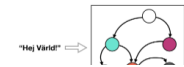
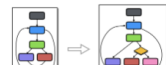
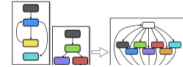
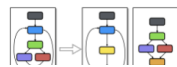
Jit
Turn a function into one that generates its code at runtime.



JitDynamic
Turn a function into one that continuously transforms itself.



Flatten
Remove control flow from a function.

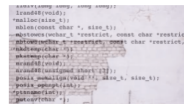


Obfuscator	Programmers	Lang	Results w/o Obfuscation	Results w/ Obfuscation
Tigress	20	C	93.65%	58.33%
Tigress	20	C	95.91%	67.22%

```
x=6;
y=7;
z=x*y;
print z;
```

```
x=E1(6);
y=E1(7);
z=x*y;
print D1(z);
```

$$x+y = \begin{cases} x-y-1 \\ (x\otimes y)+2\cdot(x\wedge y) \\ (x\vee y)+(x\wedge y) \\ 2\cdot(x\vee y)-(x\otimes y) \end{cases}$$



```
call bf
jmp L => void bf() {
    ra ← L;
    return;
}
```

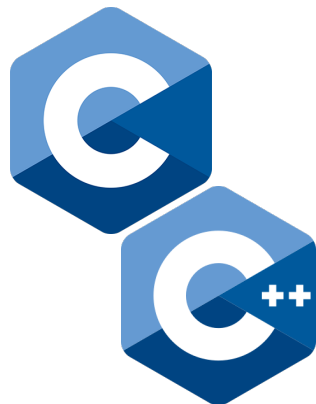
Encode Data
Replace integer variables with different representations.

Enc. Arithmetic
Replace integer arithmetic with more complex expressions.

Encode External
Hide API calls such as system calls and library calls.

Encode Branches
Make it harder to determine the target of branches.

Generalization - Python



VS



Lang.	Programmers	Classification	IG	Top-5	Top-5 IG
Python	23	87.93%	79.71%	99.52%	96.62
Python	229	53.91%	39.16%	75.69%	55.46

Takeaways

If the given codes are



...from difficult tasks,



...from skilled programmers

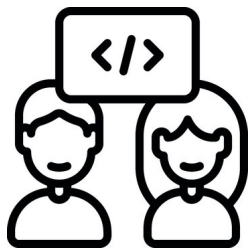


Easy to attribute

Limitation

- Multi-authorship problem

- Pair programming
- Knowledge sharing
- Generative AI

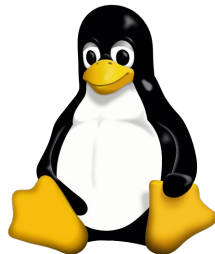


stack
overflow



- Coding style normalized problem

- Layout policy
- Layout specification
- Linter



PEP 8

©2008 - 2014

Conclusion

Conclusion

- **First work** that utilize **stylometry features** for source code stylometry
 - ...along with lexical/layout features
- Shows **>90% acc.** in classifying **1600 authors** from Google code jam
- Motivate future researches
 - Binary-only code feature set
 - Classification accuracy improving

Related works

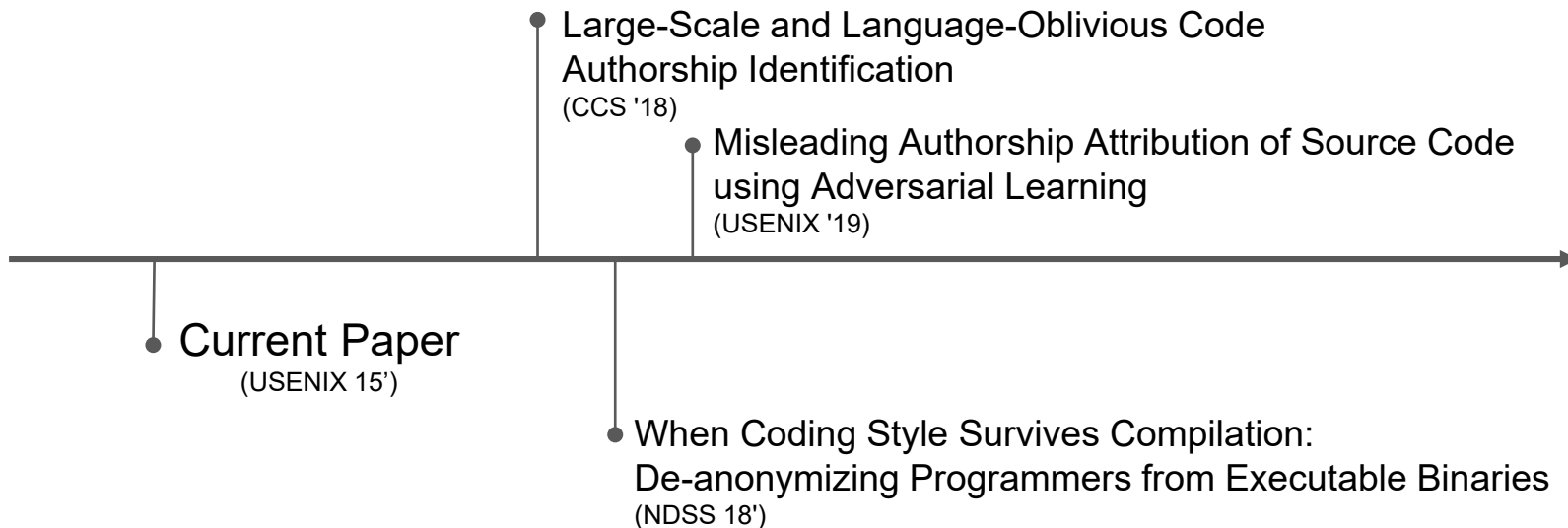
Previous, and Future works of
Code Stylometry Research

Related works - Previous works

Related Work	# of Programmers	Results
Pellin [23]	2	73%
MacDonell et al.[21]	7	88.00%
Frantzeskou et al.[14]	8	100.0%
Burrows et al. [9]	10	76.78%
Elenbogen and Seliya [11]	12	74.70%
Kothari et al. [18]	12	76%
Lange and Mancoridis [20]	20	75%
Krsul and Spafford [19]	29	73%
Frantzeskou et al. [14]	30	96.9%
Ding and Samadzadeh [10]	46	67.2%
This work	8	100.00%
This work	35	100.00%
This work	250	98.04%
This work	1,600	92.83%

Related works - Future works

Advanced Prediction Mechanism



Improved Stylometry Model

Related works - Misc

- Code similarity
 - Neural Machine Translation Inspired **Binary Code Similarity** Comparison beyond Function Pairs (NDSS 19')
 - Finding Bugs Using Your Own Code: Detecting **Functionally-similar** yet Inconsistent Code (USENIX 21')
 - How Machine Learning Is Solving the **Binary Function Similarity Problem** (USENIX 22')
- De-anonymization
 - **Deanonymization** in the Bitcoin P2P Network (ACM 17')
 - Online Website Fingerprinting: Evaluating **Website Fingerprinting Attacks** on Tor in the Real World (USENIX 22')
 - Him of Many Faces: Characterizing Billion-scale Adversarial and Benign **Browser Fingerprints** on Commercial Websites (NDSS 23')
 - Assessing **Anonymity Techniques** Employed in German Court Decisions: A De-Anonymization Experiment (USENIX 23')

Q&A

Q&A - Good Questions

- 배한성: Variable name as a feature set?
 - Could be, but might be less impactful (obfuscation, too skewed, etc)
- 이형주: Can you analyze people from different languages using just one language code?
 - Based on this paper's approaches, not straightforward (but can be generalized)
- 오성룡: Effective method against obfuscation?
 - It depends

Q&A - Best Questions

- 이승현: What is the significance of the threshold level "15%"? Is this still meaningful for $N \neq 30$? What statistical guarantees that threshold levels provide?
 - Authors should justify their threshold level for other cases for meaningful results.
- 하재현: Other code stylometry feature sets than lexical, layout, and syntactic features to improve de-anonymizing?
 - TF-IDF (Term Frequency - Inverse Document Frequency), disassembly / CFG for binary, etc.
- JIN ZHIXIAN: Difficult task -> likely attributes?? Isn't it leads collaboration & source code normalization?
 - GCJ has a competitive-programming code set. The paper assumes there is a difficult task given to only one programmer.

Thank you