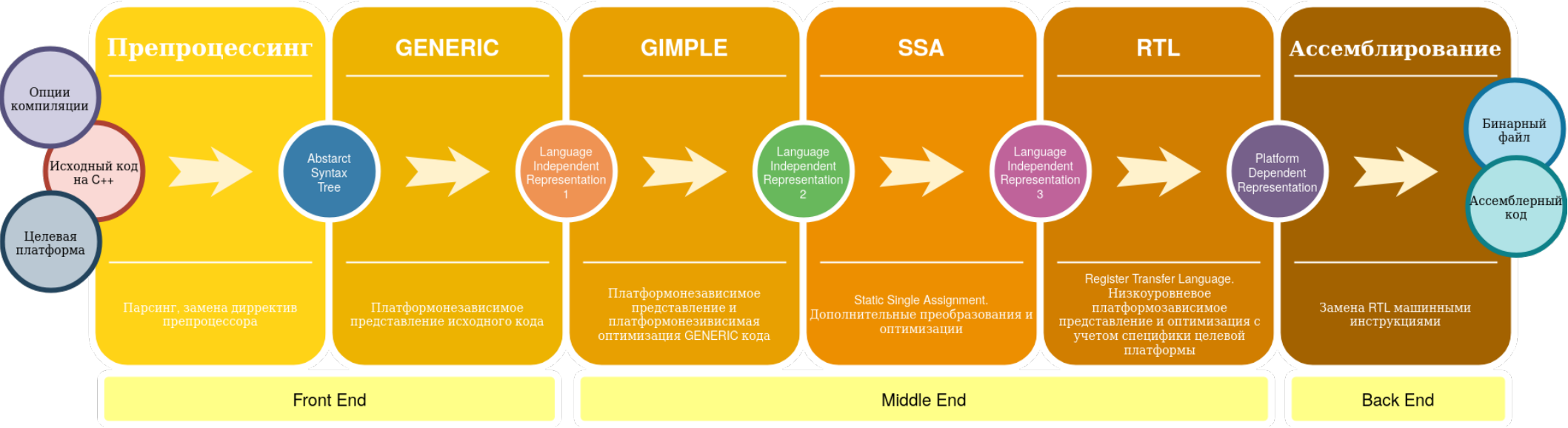
Classification of functions by ML methods based on their binary representation

24.12.2022

The essence of the task

- There is a source code of the program in a compiled programming language (PL)
- There is a binary file received after compilation
- After disassembling, there is the code of the Assembler program and the code of the functions used in the program
- **Problem**: let there be an unknown function in an unknown binary. After disassembling, the Assembler code was obtained. What does this function do?

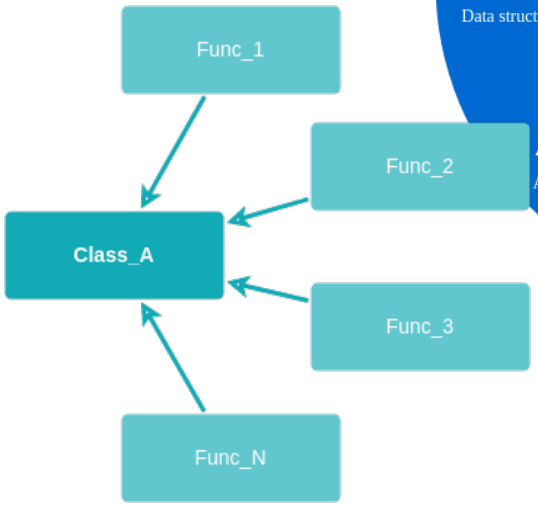
Compilation stages of GCC

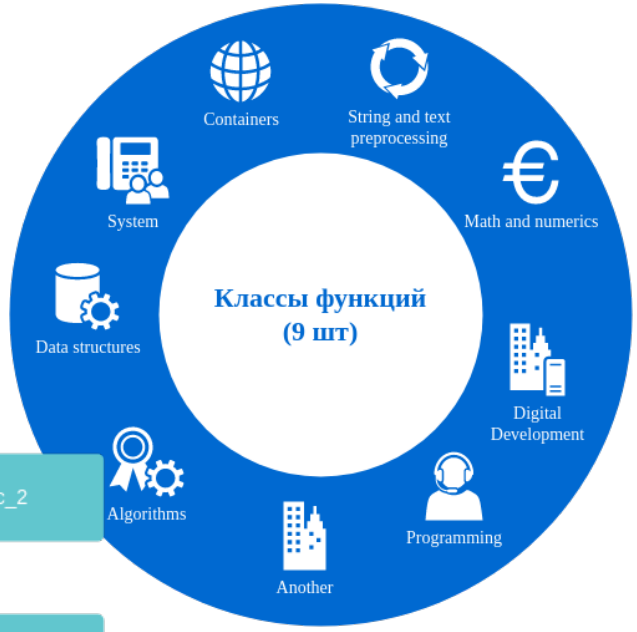


Limitations

Ограничения

1. Язык программирования
   C++
2. Тип бинарных файлов
   Linux ELF
3. Платформа
   Intel x86
4. Библиотеки
   Boost, STL, Asio, Silicon
5. Компиляторы
   GCC, Clang





Different compilers

Compiler Explorer - Mozilla Firefox

Compiler Explorer | poco/ASCIIEncoding.cpp | poco/ASCIIEncoding.h at

https://godbolt.org

git hub poco

Начальная страни...

COMPILER EXPLORER | Add... | More | Sponsors PC-lint PVS-Studio | Share | Other | Policies

C++ source #1 | Save/Load | Add new... | Vim | CppInsights | Quick-bench | C++

```
1
2       /// 7-bit ASCII text encoding.
3   class  ASCIIEncoding
4   {
5   public:
6       ASCIIEncoding();
7       ~ASCIIEncoding();
8       const char* canonicalName() const;
9       int convert(const unsigned char* bytes) const;
10      int convert(int ch, unsigned char* bytes, int length) const;
11      int queryConvert(const unsigned char* bytes, int length) const;
12      int sequenceLength(const unsigned char* bytes, int length) const;
13
14  private:
15      static const char* _names[];
16  };
17
18  int ASCIIEncoding::convert(int ch, unsigned char* bytes, int length) cons
19  {
20      if (ch >= 0 && ch <= 127)
21      {
22          if (bytes && length >= 1)
23              *bytes = (unsigned char) ch;
24          return 1;
25      }
26      else return 0;
27  }
```

x86-64 gcc 10.2 (Editor #1, Compiler #1) C++ | x86-64 gcc 10.2 | Compiler options...

```
1   ASCIIEncoding::convert(int, unsigned char*, int) c
2           push    rbp
3           mov     rbp, rsp
4           mov     QWORD PTR [rbp-8], rdi
5           mov     DWORD PTR [rbp-12], esi
6           mov     QWORD PTR [rbp-24], rdx
7           mov     DWORD PTR [rbp-16], ecx
8           cmp     DWORD PTR [rbp-12], 0
9           js      .L2
10          cmp     DWORD PTR [rbp-12], 127
11          jg      .L2
12          cmp     QWORD PTR [rbp-24], 0
13          je      .L3
14          cmp     DWORD PTR [rbp-16], 0
15          jle     .L3
16          mov     eax, DWORD PTR [rbp-12]
17          mov     edx, eax
18          mov     rax, QWORD PTR [rbp-24]
19          mov     BYTE PTR [rax], dl
20  .L3:
21          mov     eax, 1
22          jmp     .L4
23  .L2:
24          mov     eax, 0
25  .L4:
26          pop     rbp
27          ret
```

Output (0/0) x86-64 gcc 10.2 - cached (8408B)

x86-64 clang 10.0.1 (Editor #1, Compiler #2) C++ | x86-64 clang 10.0.1 | Compiler options...

```
1   ASCIIEncoding::convert(int, unsigned char*, int) const:
2           push    rbp
3           mov     rbp, rsp
4           mov     qword ptr [rbp - 16], rdi
5           mov     dword ptr [rbp - 20], esi
6           mov     qword ptr [rbp - 32], rdx
7           mov     dword ptr [rbp - 36], ecx
8           cmp     dword ptr [rbp - 20], 0
9           jl      .LBB0_6
10          cmp     dword ptr [rbp - 20], 127
11          jg      .LBB0_6
12          cmp     qword ptr [rbp - 32], 0
13          je      .LBB0_5
14          cmp     dword ptr [rbp - 36], 1
15          jl      .LBB0_5
16          mov     eax, dword ptr [rbp - 20]
17          mov     rcx, qword ptr [rbp - 32]
18          mov     byte ptr [rcx], al
19  .LBB0_5:
20          mov     dword ptr [rbp - 4], 1
21          jmp     .LBB0_7
22  .LBB0_6:
23          mov     dword ptr [rbp - 4], 0
24  .LBB0_7:
25          mov     eax, dword ptr [rbp - 4]
26          pop     rbp
27          ret
```

Output (0/0) x86-64 clang 10.0.1 - 442ms (25433B)

Read the new cookie policy Compiler Explorer uses cookies and other related techs to serve you | Consent | Don't consent

Different versions of the same compiler

find_address_dispatch_gcc_9.raw

```
xor r8d, r8d
push rbx
mov r9, rdx
mov eax, r8d
mov ebx, r8d
mov ecx, r8d
mov edx, r8d
cpuid
lea r10, [0x0800009c]
add byte [rax], al
add byte [rax], al
test eax, eax
je 0x80000ce
mov eax, 1
mov ebx, r8d
mov ecx, r8d
mov edx, r8d
cpuid
and edx, 0x4000000
je 0x80000bf
mov r10, qword [0x080000bf]
and ecx, 0x80000
je 0x80000ce
mov r10, qword [0x080000ce]
mov qword [0x080000d5], r10
mov rdx, r9
pop rbx
jmp r10
```

find_address_dispatch_gcc_10.raw

```
mov r8, rsi
xor esi, esi
push rbx
mov r9, rdx
mov eax, esi
mov ebx, esi
mov ecx, esi
mov edx, esi
cpuid
lea r10, [0x0800009a]
add byte [rax], al
add byte [rax], al
test eax, eax
je 0x80000c9
mov eax, 1
mov ebx, esi
mov ecx, esi
mov edx, esi
cpuid
and edx, 0x4000000
je 0x80000ba
mov r10, qword [0x080000ba]
and ecx, 0x80000
je 0x80000c9
mov r10, qword [0x080000c9]
mov qword [0x080000d0], r10
mov rdx, r9
mov rsi, r8
pop rbx
jmp r10
```

Getting raw data

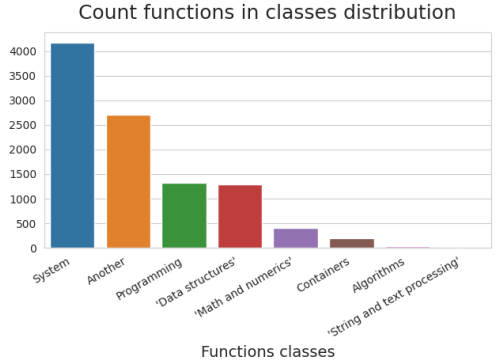
- Compilation of the Boost library by various versions of gcc compilers (10.2.1, 9.3.0) with various compilation options (O0, O1, O2, O3, Os, Og, Ofast) using the b2 build system
- Disassembling the received files using Radare2 (r2)
- Extraction of function names, number of instructions, Assembly code, block sequence graph in gml format

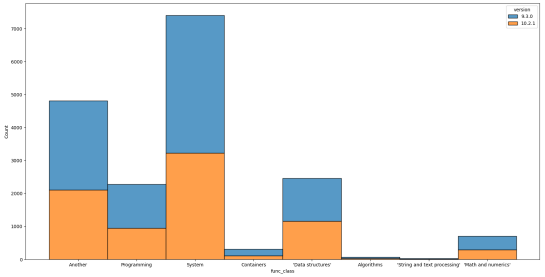
142438 lines

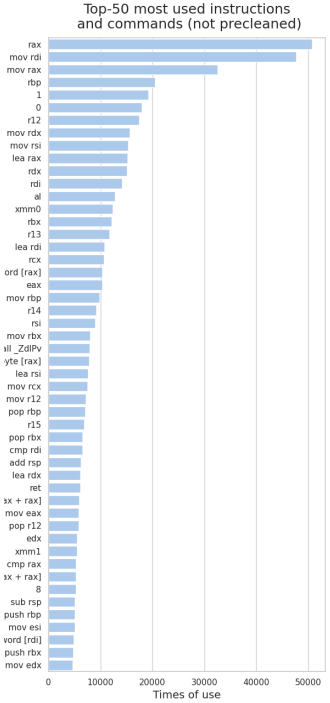
10213 functions

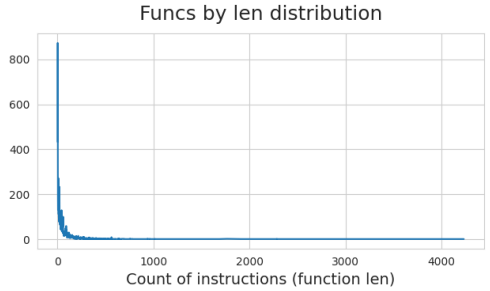
9 classes

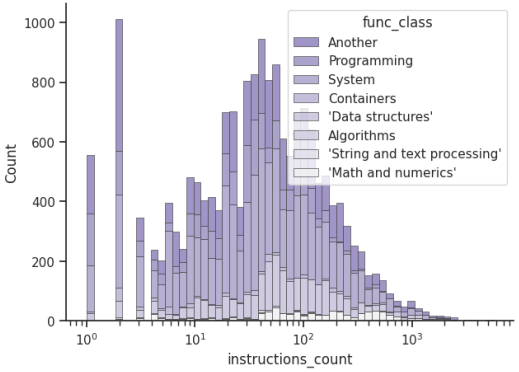
EDA

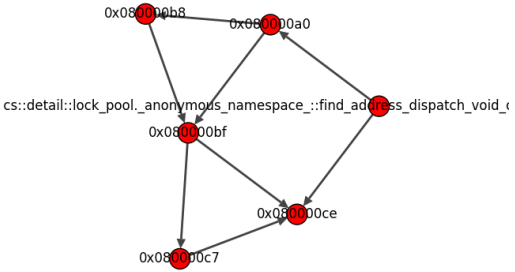

Count functions in classes distribution
4000
3500
3000
2500
2000
1500
1000
500
0
System
Another
Programming
'Data structures'
'Math and numerics'
Containers
Algorithms
'String and text processing'
Functions classes



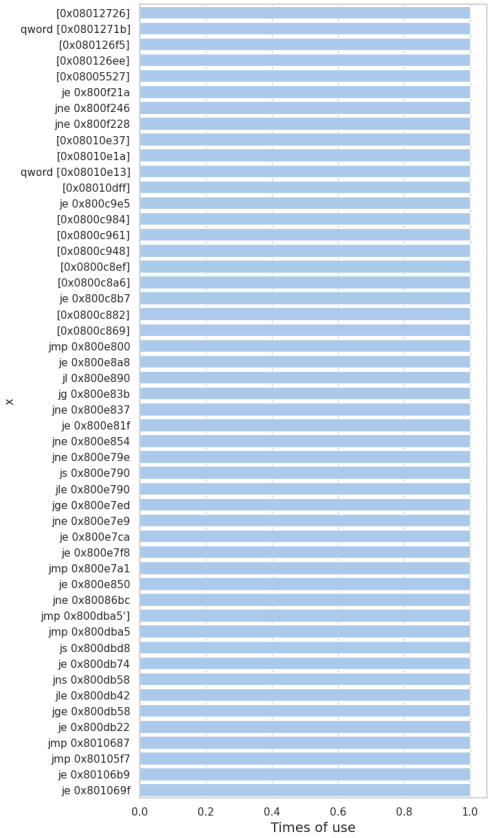

Top-50 most used instructions
and commands (not precleaned)
rax
mov rdi
mov rax
rbp
1
0
r12
mov rdx
mov rsi
lea rax
rdx
rdi
al
xmm0
rbx
r13
lea rdi
rcx
ord [rax]
eax
mov rbp
r14
rsi
mov rbx
ll _ZdlPv
yte [rax]
lea rsi
mov rcx
mov r12
pop rbp
r15
pop rbx
cmp rdi
add rsp
lea rdx
ret
ax + rax]
mov eax
pop r12
edx
xmm1
cmp rax
ax + rax]
8
sub rsp
push rbp
mov esi
word [rdi]
push rbx
mov edx
0
10000
20000
30000
40000
50000
Times of use


Funcs by len distribution
800
600
400
200
0
0
1000
2000
3000
4000
Count of instructions (function len)


1000
800
600
400
200
0
Count
10^0
10^1
10^2
10^3
instructions_count
func_class
Another
Programming
System
Containers
'Data structures'
Algorithms
'String and text processing'
'Math and numerics'


0x080000b8
0x080000a0
cs::detail::lock_pool._anonymous_namespace_::find_address_dispatch_void_
0x080000bf
0x080000ce
0x080000c7

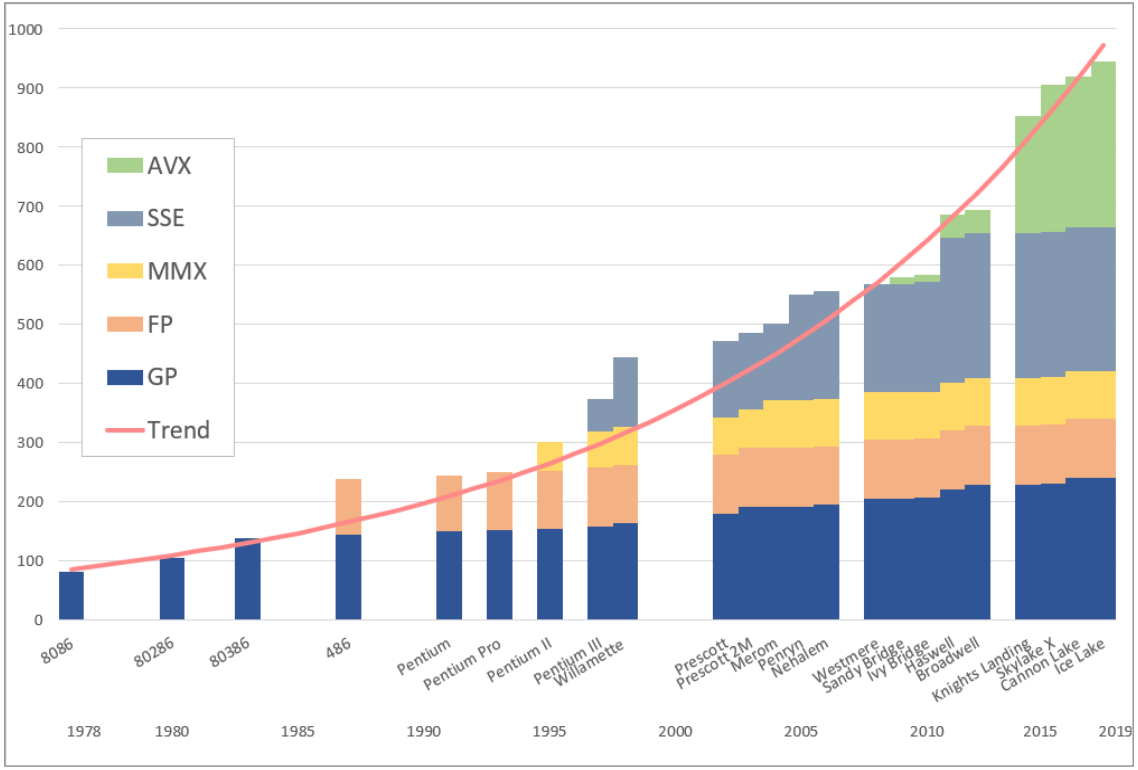
EDA



230000

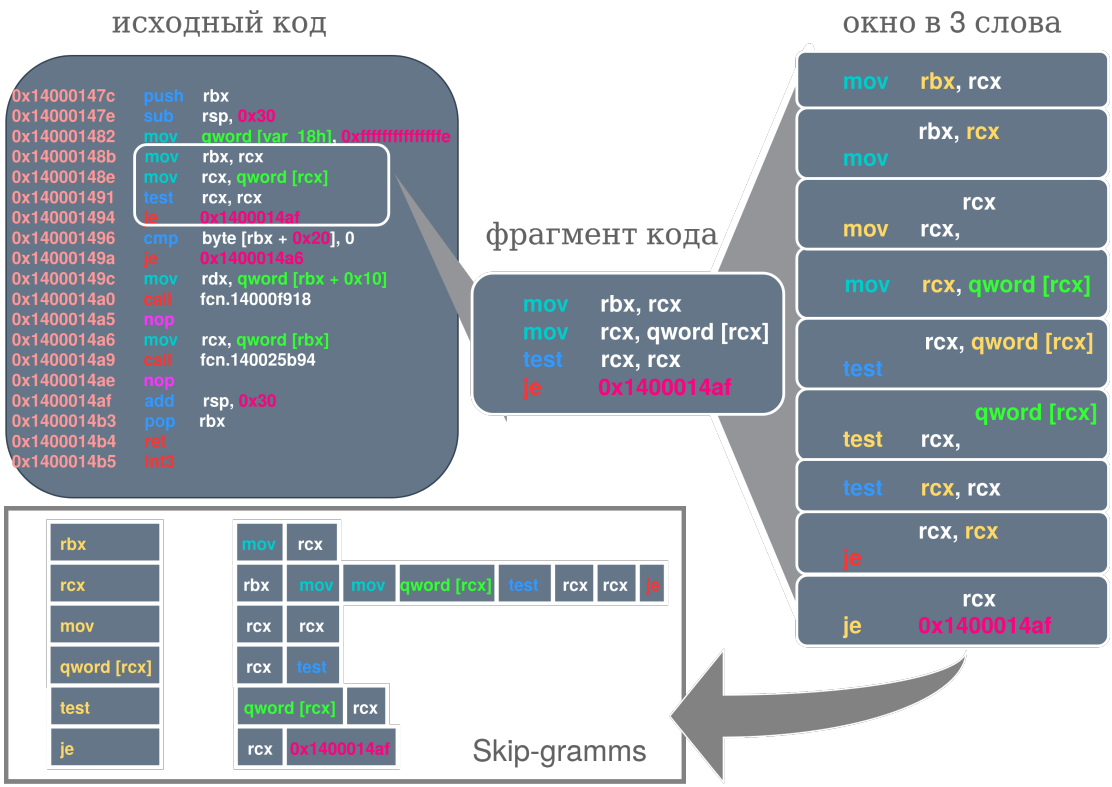
6000

~18000



https://habr.com/ru/post/503486/

CountVectorizer and TfidfVectorizer



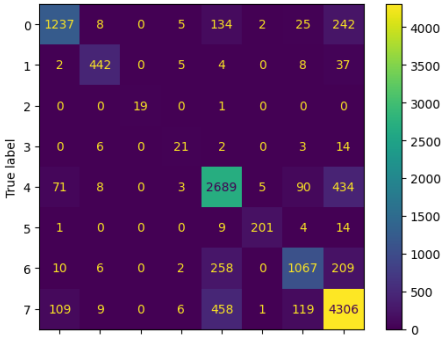
Eli5

LogisticRegression with CountVectorizer

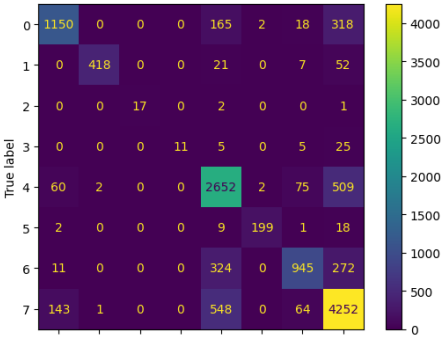
y='Data structures' top features		y='Math and numerics' top features		y='String and text processing' top features		y=Algorithms top features		y=Another top features		y=Containers top features		y=Programming top features		y=System top features	
Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature	Weight[?]	Feature
+1.352	and rdi	+1.555	st	+0.640	lock	+0.880	lea	+1.845	<BIAS>	+1.528	qword addr	+1.600	ret	+2.378	<BIAS>
+1.350	and rdi addr	+1.453	loc	+0.579	jne addr	+0.822	rax addr	+1.526	rax addr push	+1.435	sym	+1.265	<BIAS>	+2.352	rax call qword
+1.247	rbp rax call	+1.232	xmm1	+0.553	r8d	+0.749	esi	+1.378	rsi qword rbx	+1.356	boostfcn	+1.134	addr mov	+1.504	byte rax je
+1.214	rax and	+1.076	fcn call fcn	+0.538	ecx	+0.633	lea rsi	+1.311	rdi qword addr	+1.091	cmp qword		dword	+1.225	fcn nop ret
+1.212	fcn lea rax	+1.073	rsp addr	+0.535	add	+0.597	rdi qword rbp	+1.213	ret	+1.056	add rdi	+1.067	rax qword rsi	+1.216	fcn lea rsi
+1.063	mov edx dword	+1.023	fld	+0.524	jne	+0.593	rax addr ret	+1.196	addr jmp fcn	+1.028	jmp boostfcn	+1.057	close	+1.166	addr rax mov
+1.030	methodmethod	+0.969	ja	+0.520	jne addr mov	+0.551	r14 mov	+1.189	rsp call methodmethod	+1.028	jmp boostfcn nop	+1.049	rdi addr lea	+1.150	addr rax lea
+0.983	movsx	+0.944	fcn call	... 1790 more positive ...		+0.544	r14					+1.045	jne addr add	+1.107	add rdi addr
+0.935	call text	+0.921	comisd	... 63724 more negative ...		+0.542	al	+1.143	rax rax lea	+1.016	call sym	+1.027	fcn test rax	+1.102	rdi addr rcx
... 19786 more positive ...		+0.878	sub rsp addr	-0.572	addr jmp	+0.518	qword rbp	+1.124	je addr lock	+1.004	boostfcn nop	+1.001	addr add rsp	... 28143 more positive ...	
... 45728 more negative ...		+0.845	xword	-0.582	push	... 1541 more positive ...		... 16569 more positive ...		+1.004	boostfcn nop word	+0.980	word	... 37371 more negative ...	
-0.912	je addr lock	... 10419 more positive ...		-0.584	eax ret	... 63973 more negative ...		... 48945 more negative ...				+0.972	jmp fcn	-1.152	al je addr
-0.982	lea rsi addr	... 55095 more negative ...		-0.664	jmp fcn	-0.555	qword	-1.235	varfcn lea rsi	+0.990	dword addr	... 12273 more positive ...		-1.165	rdi rbp mov
-0.989	rsi qword rbx	-0.848	movq	-0.827	eax	-0.709	rsp	-1.248	rdi rax call	+0.916	rsi rdi	... 53241 more negative ...		-1.166	lea rdi addr
-1.038	rdx addr add	-0.849	qword rax	-1.094	fcn	-0.811	ret	-1.252	rax rax call	+0.894	qword addr ret	-1.036	je addr lea	-1.345	dword addr
-1.043	fcn lea rdi	-0.884	cmp	-1.548	ret	-0.910	jmp	-1.264	jmp addr nop	... 6809 more positive ...		-1.042	or	-1.411	rdi addr lea
-1.330	rdx qword varfcn	-0.925	movdqa	-3.077	<BIAS>	-2.221	<BIAS>	-1.366	rax call qword	... 58705 more negative ...		-1.077	addr add byte	-1.838	rax call fcn
								-1.501	byte rdi addr	-0.995	edi	-1.159	rdi addr test		
										-1.583	byte				

LogisticRegression vs RandomForestClassifier with CountVectorizer vs TfidfVectorizer

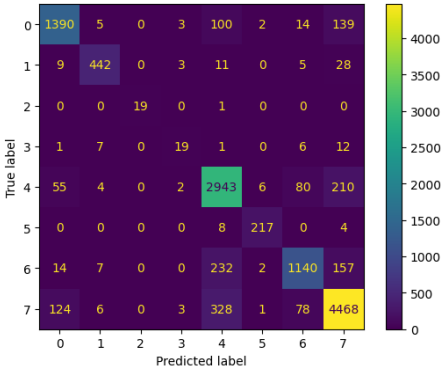
F1 - 0.81114



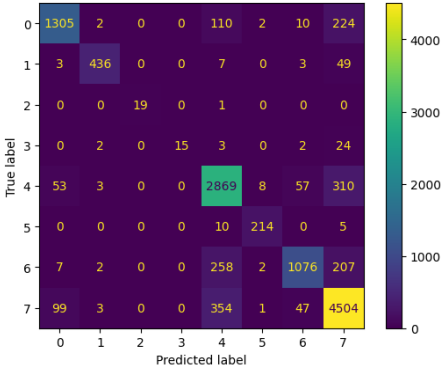
F1 - 0.783682



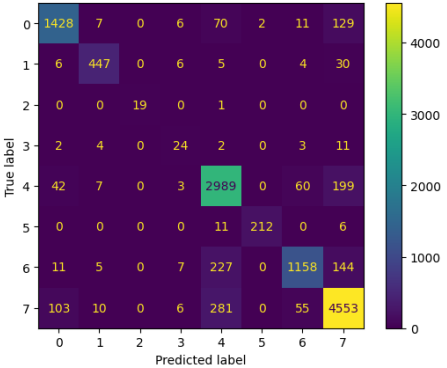
F1 - 0.864456



F1 - 0.848204



F1 - 0.88005



LogisticRegression	CountVectorizer	1-2
LogisticRegression	TfidfVectorizer	1-2
LogisticRegression	CountVectorizer	1-3
LogisticRegression	TfidfVectorizer	1-3
RandomForestClassifier	CountVectorizer	1-3

Feature extraction

```
print(*g.get_adjacency())
print(g.vcount())
print(g.incident(0))
print(g.degree(0))
print(g.diameter())
print(g.girth())
print(g.radius())
print(g.average_path_length())
print(g.transitivity_avglocal_undirected())
print(g.laplacian())
```

- Working with a graph (python grapl
- Working separately with operands and operators
- Working with addresses
- Prologue and epilogue of functions
- Stack
- Optimization options
- More compilers and their versions
- More libraries

Where it can be used, in what form to implement

- Traffic filter plugin in IDS Suricata
- Plugin for Radare2 for the initial analysis of unknown functions

End