

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Specializarea: Inteligență artificială

LUCRARE DE DISERTAȚIE

Coordonator științific:
Conf. Dr. Traian ȘERBĂNUȚĂ

Absolvent:
Teodor Mircea POPESCU

București
2019

UNIVERSITATEA DIN BUCUREȘTI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Specializarea: Inteligență artificială

Sistem Online de Modelare
Orientat pe Neuronii
(S.O.M.O.N.)

Coordonator științific:
Conf. Dr. Traian ȘERBĂNUȚĂ

Absolvent:
Teodor Mircea POPESCU

București
2019

Cuprins

INTRODUCERE	1
1. DESCRIEREA GENERALĂ A PLATFORMEI.....	2
2. DESPRE PROIECT.....	3
2.1. POVESTEA	3
2.2. NOTAȚII	5
3. MODUL DE OPERARE (WEBSITE)	6
3.1. FUNCTIONS/DEFAULT FUNCTIONS	6
3.2. REQUESTS	6
3.2.1. Functions/Requests	6
3.2.2. Functions/Requests/View solutions	6
3.3. FUNCTIONS /PUBLIC FUNCTIONS	7
3.3.1. Simplify	11
3.3.2. Guess function	13
3.4. INSTANȚE.....	16
3.4.1. Add instances	16
3.4.2. Automated instances	16
3.4.3. View instances	16
3.5. CLASE	18
3.6. VIEW GRAPHS.....	19
3.7. USERS.....	21
3.8. FUNCTIONS STATISTICS.....	21
3.9. SETTINGS	22
3.10. DOWNLOAD.....	22
4. MINER.....	23
4.1. DEFAULT.....	23
4.2. PUBLIC	26
4.3. VARIABLE	26
4.4. CONSTANT	26
4.5. IF	27
4.6. CLASS	27

4.7. CLASSGETTER	27
4.8. RECURSIVE	27
5. BAZA DE DATE.....	28
6. EXEMPLE	30
6.1. EXEMPLE CU FUNCȚII EXISTENTE	30
6.1.1. Basic	30
6.1.2. File transform	30
6.1.3. Inteligența artificială	31
6.2 EXEMPLE ȘI FUNCȚII PE CARE NU LE-AM IMPLEMENTAT ÎNCĂ	45
6.2.1. Basic	45
6.2.2. File transform	45
6.2.3. Inteligența artificială	45
7. PROVOCĂRI.....	47
8. PROGRAME SIMILARE	48
9. DIRECȚII DE ÎMBUNĂTĂȚIRE	50
9.1. UTILIZAREA SC	50
9.2. ORDINEA EXECUTĂRII.....	50
9.3. TRANSLATOR PENTRU ALTE LIMBAJE	50
9.4. VARIANTĂ LOCALĂ	50
9.5. OPTIMIZĂRI	51
9.6. SECURITATE.....	51
9.7. PARALELIZABIL	51
9.8. SETĂRI	51
9.9. HIGH ORDER FUNCTIONS.....	51
9.10. MAI MULTE STATISTICI.....	52
9.11. SIMPLIFICAREA UTILIZĂRII.....	52
9.12. ERORI.....	52
9.13. TIPURI NOI DE DATE	53
9.14. ALTELE	53
10. CONCLUZII.....	54
11. SURSE FOLOSITE	61

Introducere

Această lucrare reprezintă o continuare firească a lucrării de licență „M.A.C.R.O.U.L.”. Acolo vorbeam despre modul în care multe activități sunt repetitive și merită automatizate. Făcusem mai multe funcții pe care le tot foloseam și de fiecare dată, pentru orice proiect nou, câteva funcții se tot repetau. Am făcut astfel un mini limbaj de programare cu 14 instrucțiuni simple (Click, Muta, Asteapta, Taste, Executa, Website, Email, Power, SSH, Inchide, Redimensionare, Asteapta_schimbare, Macrou, Sablon) care sunt suficiente pentru automatizarea foarte multor rutine (am menționat și atunci că Click și Taste sunt suficiente deoarece noi le folosim doar pe acestea două de cele mai multe ori). Încă folosesc Macroul, dar are unele limitări. De exemplu, lipsa variabilelor. Am adăugat și această posibilitate (sub formă de șabloane) dar devenea prea greu de folosit. Acele variabile erau necesare (de cele mai multe ori) pentru transformarea datelor. Astfel, am adunat suficiente idei pentru următorul mare proiect: Somonul.

O altă asemănare cu lucrarea de licență este reprezentată de dorința de a face programarea cât mai simplă, astfel încât să fie accesibilă oricui. Un limbaj care să fie suficient de simplu astfel încât după vizualizarea unui exemplu să fie oricine în stare să folosească platforma.

Modul de programare a rămas asemănător cu cel de acum zece-douăzeci de ani. Asta nu este neapărat un lucru rău, dar există motive pentru care au fost înlocuite limbajul de asamblare, codul mașină etc. cu C, C++, Java etc.

Avantajele principale ale schimbării spre un limbaj mai înalt sunt:

- Sintaxa este mai ușor de înțeles;
- Viteza de scriere a codului mai mare (și evident viteza de schimbare a acestuia);
- Timpul de învățare este mai mic;
- Chiar dacă sunt știute ambele sintaxe, tot este mai ușor să se înțeleagă despre ce este vorba în limbajul mai înalt. Un exemplu la care o sa revin este „quicksort”-ul în Haskell.

```
qs [] = []
```

```
qs (x:xs) = qs (filter (< x) xs) ++ [x] ++ qs (filter (>= x) xs)
```

Cod 1

Dezavantajele sunt:

- Eficiența;
- Pierderea posibilității de a alege cum se execută codul (programul în Haskell de mai sus nu are sortare in-place).

Astfel, am ales să introduc un nivel nou de abstractizare, peste programele scrise deja în diverse limbaje (sau pentru operații simple, precum adunarea, să le implementez eu în C++), în care modul de a programa este online (deci se poate folosi aproape orice dispozitiv modern) folosind grafuri.

1. Descrierea generală a platformei

Am ales ca principalul mod de a interacționa cu această platformă să fie prin intermediul unui browser deoarece este cel mai simplu și la îndemână de accesat. O aplicație similară se putea face (poate chiar mai bine) doar pe Windows, dar ar limita mult accesul la un limbaj de programare care este făcut „pentru toți”.

Pe acest site (în principal) se pot face:

- funcții noi, folosind funcții implicite („default”) sau funcții implementate de către alți utilizatori (publice);
- clase (ce pot fi utilizate ulterior în funcții);
- instanțe (funcții ce urmează să fie rezolvate);
- descărcarea miner-ului (program executabil în Windows ce rezolvă instanțele);
- cereri pentru funcții: fie publice (dacă se pot rezolva folosind funcții existente în acel moment), fie default (pentru adăugarea de funcționalități noi de bază).

Funcționalitatea se poate extinde și prin folosirea funcției default JsOnUrl ce permite executarea unui cod Javascript/JQuery pe un anumit site. De exemplu „\$(\".r\")[0].innerHTML” pe site-ul „<https://bitcoinwisdom.com/>” are ca rezultat Stringul "6432.27" și are ca semnificație prețul unui Bitcoin (la acest moment) în dolari. Dar folosirea acestei funcții nu este încurajată (din cauză că site-urile pot să își schimbe conținutul).

Instanțele (și datele acestora) sunt implicit destinate să fie rezolvate de către cel care a făcut instanța și utilizatorul rezolvă implicit doar instanțele puse de el (se poate schimba din setări să rezolvi instanțele altora și să adaugi instanțe ce pot fi rezolvate de către oricine). Acest lucru se întâmplă deoarece datele necesare rezolvării unei instanțe pot avea caracter personal. Iar această setare există deoarece acele date (fișiere) ajung din surse posibil necunoscute sau pot ajunge pe calculatoare necunoscute. Dacă se dorește, din moment ce platforma este open-source, se poate crea un server care poate rula MySQL și PHP. Astfel, datele pot fi cât de sigure dorește utilizatorul.

Arhitectura este cea clasică, pe 3 niveluri:

- Baza de date MySQL;
- Serverul PHP ce furnizează datele către clienți;
- Aplicații clienți (site și miner).

2. Despre proiect

2.1. Povestea

Această lucrare reprezintă o colecție de aplicații (pe care le credeam independente) la care m-am gândit în ultimii ani.

Prima idee a fost aceea că este nevoie de a vizualiza tot codul sau măcar părți din el. De multe ori sunt folosite scheme logice pentru a arăta cum funcționează un program. M-am gândit că o aplicație interesantă ar fi una care primește o bucată de cod, o parsează și o reprezintă ca un graf. Problema la această idee este scalabilitatea. Chiar dacă aș sta (probabil un timp considerabil) pentru a face asta chiar și într-un limbaj simplu (cum este C de exemplu), acel limbaj prezintă riscul să nu mai fie folosit și acest proces trebuie luat de la capăt pentru alte limbaje. Așa că m-am gândit că poate e o idee bună să pot programa direct folosind scheme logice. Căutând un exemplu de schemă logică am găsit imaginea din Figura 1 [12] și am transformat schema logică în funcția din Figura 2. Am folosit în funcția creată doar variabile, constante și funcții implicite (nodurile „Mod” și „==0” pot forma împreună o funcție „Divide” iar cele două if-uri împreună cu nodul „Not” ar putea fi comprimate într-o funcție If-then-else).

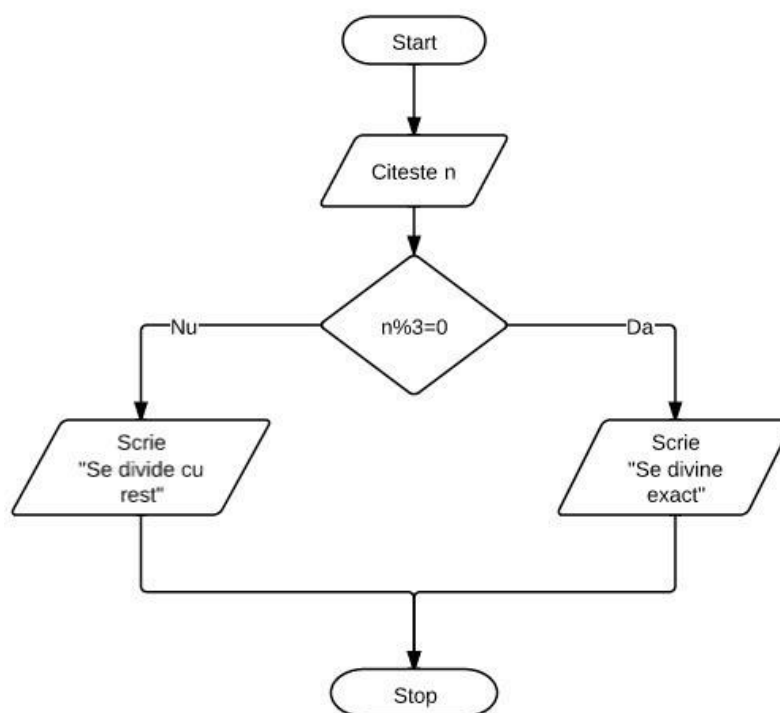


Figura 1

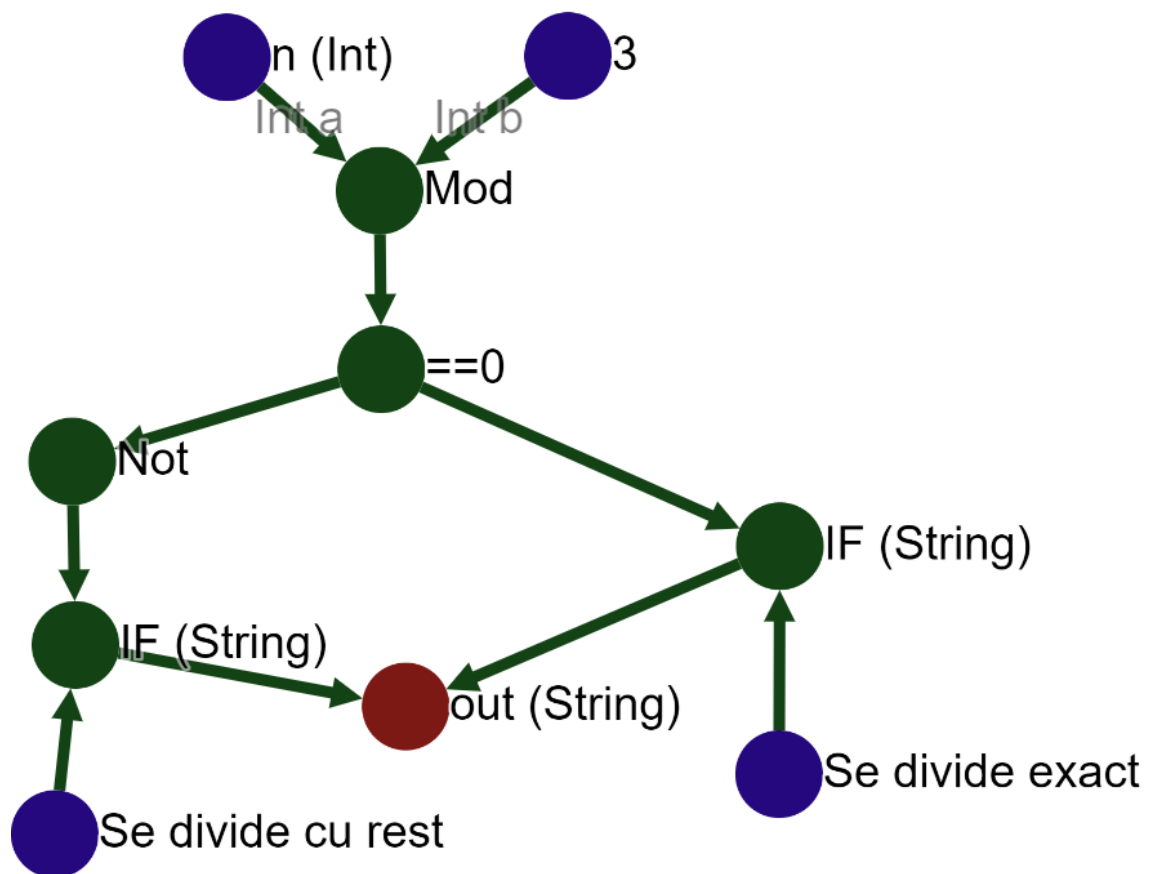


Figura 2

O altă idee a fost aceea de a face funcții publice, accesibile tuturor. În acest mod, se pot folosi funcții deja create într-o funcție mai mare (pentru a construi funcția dorită mai repede și mult mai clar), pentru a fi folosite direct (fără să mai fie implementată aceeași funcție de două ori) în instanțe, pentru a învăța din funcția respectivă noi moduri de a face funcții etc. Există deja multe bucăți de cod care se găsesc pe internet pentru diferite probleme (snippets) dar este posibil să fie în alt limbaj de programare sau chiar dacă este în limbajul dorit, de multe ori mi s-a întâmplat să fie nevoie de librării suplimentare ce necesită timp în plus de instalare și crește complexitatea proiectului (și dependențele acestuia).

Având astfel o listă structurată de funcții, deja apar avantaje față de alte limbaje. Având un nod (neinițializat) care primește ca input un nod variabilă String și se duce într-o variabilă de tip Bool, dacă se alege ca acel nod să devină o funcție default, atunci are o singură opțiune din 39 funcții default (==String care compară două String-uri). În alte limbaje precum C++ nu există opțiunea de a „cere” funcția care primește un String și returnează un Bool (chiar dacă sunt mai puțin de 10 astfel de funcții în sute de fișiere). Această funcționalitate este implementată sub numele de „Guess function”.

O altă idee a fost aceea de a utiliza resursele mai eficient. Sunt multe calculatoare vechi care nu merită să fie folosite la minat Bitcoin sau alte criptomonede, dar sunt suficient de bune pentru prelucrarea unor imagini (resize, OCR) de transformare a unor fișiere (din .mp3 în .wav,

din .pdf în listă de imagini etc.), operațiuni utile și în domeniul inteligenței artificiale (de exemplu aducerea imaginilor la dimensiuni mici de tipul 32x32 sau 64x64 pixeli pentru a fi prelucrate ulterior). Există aplicații/site-uri ce au nevoie de putere computațională pentru o perioadă scurtă de timp (afișarea rezultatelor la BAC sau alte examene, Black Friday) mai mare față de restul anului. Acestea ar putea beneficia de un aranjament de ajutor reciproc și ar elimina nevoia cumpărării mai multor servere.

Pornind de la aceste idei principale (vizualizarea codului, funcții structurate și publice, computație distribuită) am început dezvoltarea unui program ce permite cele de mai sus, la care am adăugat și alte funcționalități relevante pentru proiect.

După ce am notat ce am vrut să construiesc, am verificat dacă mai există ceva asemănător. Articole despre „Data flow” există încă din 1974 [1] dar și câteva implementări precum Tensorflow sau LabVIEW [2]. Dar acestea nu au toată generalitatea pe care o propun eu (LabVIEW este folosit pentru dispozitive hardware, senzori, electronice etc. iar Tensorflow este folosit pentru machine learning). Alte programe nu sunt gratuite/open source sau nu sunt distribuite etc.

2.2. Notății

În modul de folosire a platformei și în detaliile de implementare o să folosesc următoarele notații:

Funcție default/implicită = Funcție de bază care este implementată deja în miner. De exemplu +, -, Mod, Div, Int2Str etc.

Funcție publică = Funcție creată de către utilizator, formată din mai multe noduri.

Instanță = Pereche formată din o funcție publică și date de intrare. De exemplu, funcția publică „Even” cu datele de intrare „Int n=7,.. Aceasta are trei stări principale (în așteptare, în rezolvare și rezolvat).

Miner = Executabil ce permite rezolvarea unor instanțe.

Tipuri de date = Int, String, Bool, Image etc.

Tipuri de noduri = Variable, Constant, IF, Default, Public, Recursive, Class, ClassGetter, Filter.

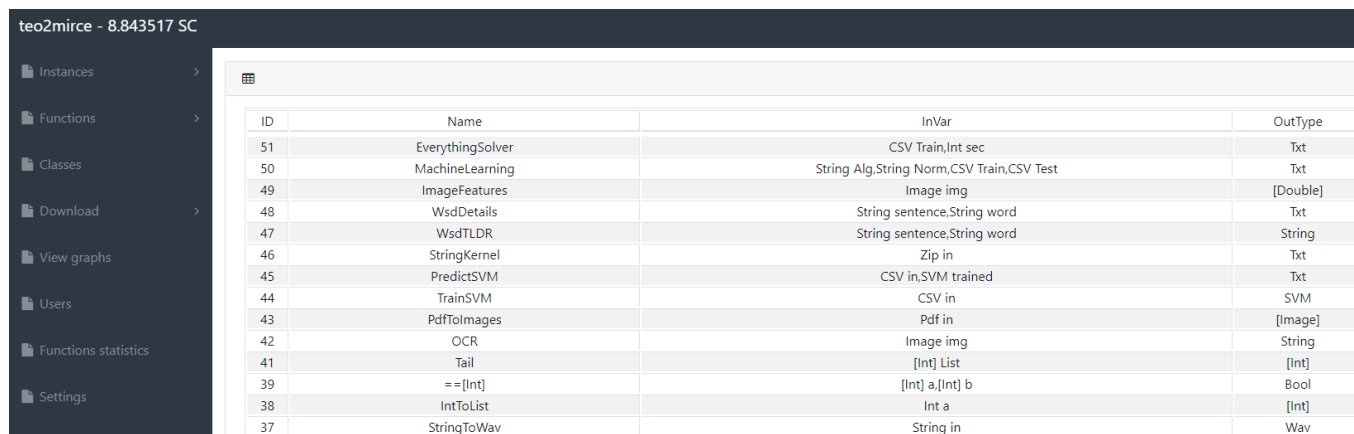
Neuron = Nod = Funcție care are zero, unul sau mai multe input-uri de la alte noduri și poate returna un rezultat de un anumit tip către alte noduri. De exemplu funcția default „==0” primește un Int și returnează un Bool. Tipurile principale ale unui nod sunt variabile, constante, funcții publice și funcții default.

Muchie = Legătura orientată între două noduri, are o valoare (port) care indică parametrul pentru nodul destinație. De exemplu funcția „-” are două porturi de intrare („Int a” și „Int b”) și are ca rezultat a-b.

SC = Somon Coin = monedă virtuală ce se obține rezolvând instanțe și se poate folosi pe site în diferite moduri (precum sponsorizarea unei funcții publice).

3. Modul de operare (website)

3.1. Functions/Default functions



ID	Name	InVar	OutType
51	EverythingSolver	CSV Train,Int sec	Txt
50	MachineLearning	String Alg,String Norm,CSV Train,CSV Test	Txt
49	ImageFeatures	Image img	[Double]
48	WsdDetails	String sentence,String word	Txt
47	WsdTLDR	String sentence,String word	String
46	StringKernel	Zip in	Txt
45	PredictSVM	CSV in,SVM trained	Txt
44	TrainSVM	CSV in	SVM
43	PdfToImages	Pdf in	[Image]
42	OCR	Image img	String
41	Tail	[Int] List	[Int]
39	==[Int]	[Int] a,[Int] b	Bool
38	IntToList	Int a	[Int]
37	StringToWav	String in	Wav

Figura 3

În această pagină utilizatorul poate vedea ce funcții implicite sunt implementate în miner. Sunt arătate numele funcției, porturile de intrare, tipul de ieșire și eventuale detalii ajutătoare (funcția „-” are rezultatul a-b).

3.2. Requests

3.2.1. Functions/Requests

Dacă o funcție nouă nu se poate obține cu ajutorul funcțiilor implicite, sau utilizatorul nu știe să facă acea funcție, se poate face un request (o cerere) pentru rezolvarea problemei respective. Request-ul poate să fie de tip public (se poate rezolva cu funcțiile existente) sau default. Acestea pot fi votate (de către oricine) folosind SC pentru a arăta cât de importantă este problema respectivă pentru comunitate. Acestea pot avea două stări: Active și Done (rezolvate). Doar cel care a propus acel request (inițial Activ) poate să schimbe starea. În această pagină se pot vedea request-urile existente (este de preferat să nu existe duplicate) și se pot adăuga request-uri noi sau pot fi vizualizate soluțiile propuse pentru un anumit request.

3.2.2. Functions/Requests/View solutions

Selectând un request se pot vizualiza soluțiile propuse. La rândul lor, propunerile pot fi votate. În această pagină este arătată funcția publică (numele și id-ul), câți SC au fost investiți în acea propunere și cât SC are funcția publică respectivă.

3.3. Functions /Public Functions

Add/Copy function Guess function View graph Description Examples Simplify Send 1 SC						
ID	SC	Name	In variables	Out variables	Creator	Dependency
64	100	QuickSort	[Int] in	[Int] out1,[Int] out2	teo2mirce	or,>,<==Int,Head,Not,ConcatInt,IntToList,==[Int],Tail
2	5.033933332	Even	Int n	Bool out	teo2mirce	Mod,==0
84	3	todouri		Int out	teo2mirce	
65	1	Fact	Int n	Int out	teo2mirce	--,==Int,*!=Int
61	1	Prime	Int n	Bool out	teo2mirce	Mod,==0,Div,...==[Int]
59	1	NotDivides	Int n,Int b	Bool out	teo2mirce	Mod,==0,Not
54	1	IsEmpty	[Int] in	Bool out	teo2mirce	==[Int]
51	1	OCR	Image in	String out	teo2mirce	OCR
45	1	>=	Int a,Int b	Bool out	teo2mirce	or,>,<==Int
44	1	<	Int a,Int b	Bool out	teo2mirce	<
41	1	>	Int A,Int B	Bool Out	teo2mirce	>
3	1	Divides	Int n,Int b	Bool out	teo2mirce	Mod,==0
43	0.0620833333000000004	test concats	[Int] L,Int x,[Int] G	[Int] out	teo2mirce	ConcatInt,IntToList
63	0.000066666	Primes	Int limit	[Int] out	teo2mirce	Mod,==0,Div,...==[Int]
96	0	Inutil2		Int Q	teo2mirce	+
95	0	FuncieTest		Int Q	teo2mirce	+
94	0	evenIneficient	Int n	Bool out	teo2mirce	Mod,==0,>,...
93	0	Nedeterminism		Int out2,Int out3	teo2mirce	
92	0	Kernel	Zip in	Txt out	teo2mirce	StringKernel
91	0	PCV2	Int w,Int x,Int t	Int out	teo2mirce	+,ifte,>,<==Int,Div,*
90	0	PC	Int w,Int x,Int t	Int out	teo2mirce	+,ifte,>,<Div,*!=Int
89	0	OameniNoi		String out	teo2mirce	JsOnUrl
88	0	EvenHalfOddDouble	Int n	Int out	guest	Mod,==0,ifte,Div,*

Figura 4

În această pagină sunt afișate funcțiile publice. Deoarece pot ajunge să existe prea multe funcții publice, acestea sunt filtrate în funcție de setări (vezi capitolul setări, se pot afișa funcții făcute de către oricine, depășind un anumit prag de SC sau făcute de către utilizatorul logat în ultimele x ore).

Funcțiile nu pot fi editate după ce au fost făcute (din motive de securitate în principal). Pentru exemplul din introducere legat de prețul Bitcoin-ului, modificarea funcției înseamnă că se poate schimba site-ul accesat de către cei care rezolvă instanța respectivă.

Primul câmp este cel de ID. Deoarece numele funcțiilor pot să fie similare, cel mai sigur este să fie verificat ID-ul când este folosită funcția. Apoi SC arată cât de mult a fost investit în funcția respectivă. Apoi numele funcției, variabilele de intrare și de ieșire pentru funcția respectivă, cel care a creat funcția și dependențele de funcțiile default. Dependențele de funcții default sunt tot o măsură de siguranță. De exemplu funcția Even are nevoie doar de Mod și „==0”. Dacă are nevoie să acceseze un site sau o altă funcție, deja funcția devine suspectă.

Ca funcționalități, se pot vota/susține funcții donând SC, se poate vedea descrierea funcției ce a fost scrisă când a fost făcută funcția.

Se poate vedea graful funcției (cu roșu sunt nodurile de ieșire, cu albastru nodurile de intrare sau constantele). De exemplu funcția Even:

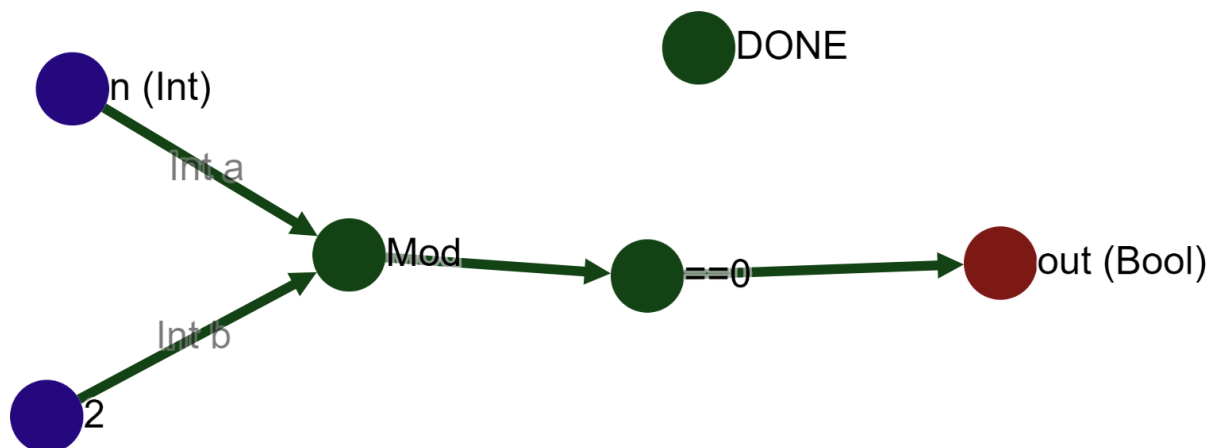


Figura 5

După ce o instanță a unei funcții a fost rezolvată, aceasta poate să devină un exemplu pentru funcția respectivă. Astfel, funcția respectivă devine mai sigură/credibilă dar și mai clară în ceea ce își propune să rezolve. Multe librării/bucăți de cod, probabil bune, au fost ignorate (de către mine cel puțin deoarece nu prezentau un exemplu simplu sau convingător).

Principala funcționalitate este adăugarea unei funcții. Dacă o funcție este selectată când se apasă pe „Add/Copy function” se continuă de acolo. Adică, se face o copie la funcția existentă și se pot face mici modificări, fără a fi nevoie să fie adăugate manual toate nodurile din funcția inițială.

În pagina de construcție a unei funcții, utilizatorul poate adăuga noduri (făcând click) având tipul de date inițial „untyped” (neinițializat). Tipul se poate schimba făcând click de două ori pe un nod. Tipurile sunt: "Variable", "Constant", "IF", "Default", "Public", "Done", "Recursive", "Class", "ClassGetter", "Filter".

Între noduri se pot face legături/muchii, făcând întâi click pe nodul sursă și apoi pe nodul destinație. Această muchie are nevoie de un port, o valoare ce reprezintă parametrul de intrare pentru funcția din nodul destinație. Dacă legătura este evidentă (înseamnă că știu tipul de date al sursei, știu ce porturi de intrare are destinația și există un singur port în destinație de tipul sursei), portul se completează automat (pentru ușurința de a programa). De exemplu, dacă sursa este un Int iar destinația este o funcție care primește un Int și un String este evident, iar dacă nodul destinație este o funcție ce primește două Int-uri, nu mai este evident.

Variabilele sunt folosite în special pentru a arăta care sunt nodurile de intrare/ieșire. Au un nume și un tip de date (Int, String, Image etc.). Portul de intrare este gol deoarece pot intra doar date de tipul respectiv, nu există ambiguitate. Pot fi folosite fără să fie noduri de intrare/ieșire dacă se dorește scoaterea în evidență a unui nod mai important (prin numele variabilei).

Constantele sunt folosite pentru a nu fi nevoie de introducerea unor variabile de intrare ce trebuie completate cu aceleași valori pentru fiecare instanță. De exemplu, pentru a face o funcție „Even” ce verifică dacă un Int este par sau impar, fără constante ar fi nevoie să fie introdus

mereu numărul 2 pe lângă numărul care trebuia verificat. Acestea nu au porturi de intrare deoarece constantele nu pot avea noduri de intrare.

IF este o funcție care lasă sau nu valorile să treacă. Când este definit acest nod, trebuie specificat tipul de date al valorii care poate trece. Ca porturi de intrare are „Tip a” (unde tip este tipul de date menționat) și „Bool cond” ce reprezintă condiția necesară ca variabila „a” să treacă mai departe.

Funcțiile default sunt cele implementate deja în miner. Structura funcțiilor este păstrată în baza de date. Sunt 39 de astfel de funcții definite (la momentul scrierii lucrării). Un exemplu foarte simplu este funcția „+” (cu ID-ul 1). Aceasta are două porturi de intrare („Int a” și „Int b”) și returnează un rezultat de tip „Int”. Lista de funcții default se găsește la pagina „Functions/Default functions”.

Funcțiile publice sunt cele făcute de către utilizatori. Porturile de intrare pentru o funcție publică sunt date de numele și tipul variabilelor de intrare ale acelor funcții. De exemplu, funcția Even are ca port de intrare „Int n”. Tipul de date de ieșire al funcției este dat de variabilele de ieșire din momentul salvării funcției (toate variabilele de ieșire trebuie să aibă același tip de date).

Funcția „Recursive” cere un tip de date. În momentul în care funcția se salvează, tipul de date al nodului de intrare trebuie să fie egal cu tipul de date al nodurilor de ieșire și acestea să fie egale cu tipul de date definit la crearea nodului de tip „Recursive”.

Tipul „Class” reprezintă un constructor al unei clase, clasa trebuie să fie definită anterior (în pagina „Classes”). Nodul are nevoie, ca informație, de numele clasei. El returnează un obiect de tipul clasei respective.

„ClassGetter” accesează și întoarce un câmp al unui obiect. Are nevoie pentru a fi definit de clasa pentru obiectul respectiv și câmpul obiectului ce se dorește a fi extras.

Filter este una dintre cele trei funcții „high order” (pe lângă „map” și „reduce”). Acest tip de nod permite filtrarea unei liste (de un anumit tip de date) folosind funcții publice deja create. Se pot folosi doar funcții ce întorc un rezultat de tip Bool (de exemplu Even). Pentru definirea nodului este nevoie de funcția publică și de portul funcției care să „evolueze” (ceea ce este diferit de alte limbaje). Adică, dacă folosim funcția Divides (Figura 6), putem fie evolua portul „Int n” fie „Int b”. Astfel, având o listă de numere L și un număr a, în funcție de port alegem să evoluăm, putem fie să vedem ce numere din L sunt divizori ai lui „a” fie ce numere din L sunt multipli ai lui a. Un alt exemplu: folosind doar funcția „>” putem filtra o listă de numere L comparând cu un număr a și să extragem numerele din L mai mari sau mai mici decât a, în funcție de portul ales pentru nodul Filter.

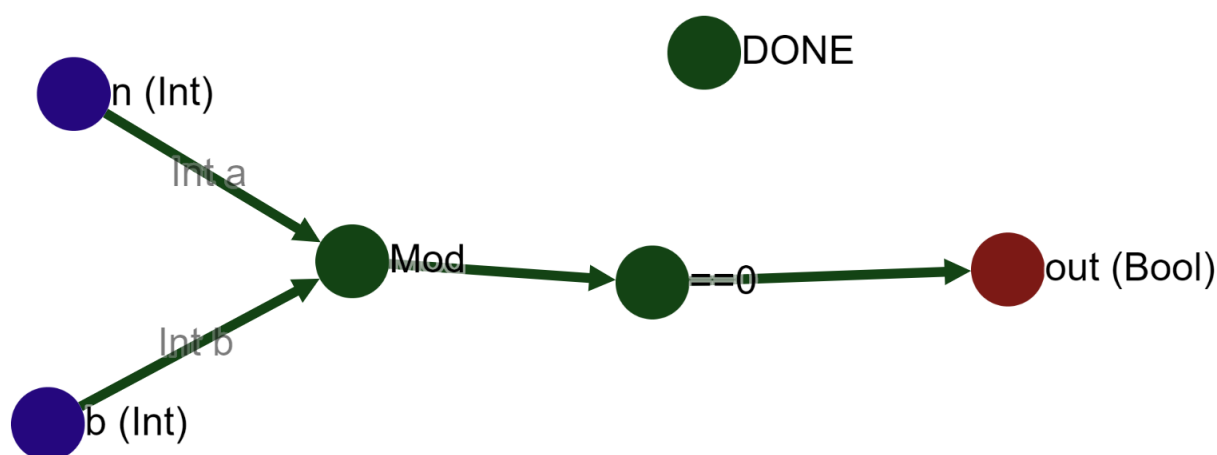


Figura 6

Apăsând pe „DONE” apare dialogul de salvare a funcției dacă sunt îndeplinite anumite condiții. Pentru a salva funcția, trebuie completat numele funcției, o scurtă descriere și apoi alese nodurile (variabilele) de intrare și cele de ieșire. Chiar dacă sunt mai multe variabile de ieșire, când se rezolvă instanța funcției respective, instanța se termină când se ajunge prima dată la un nod de ieșire. Asta înseamnă că nu este clar ce se returnează în cazul următor:

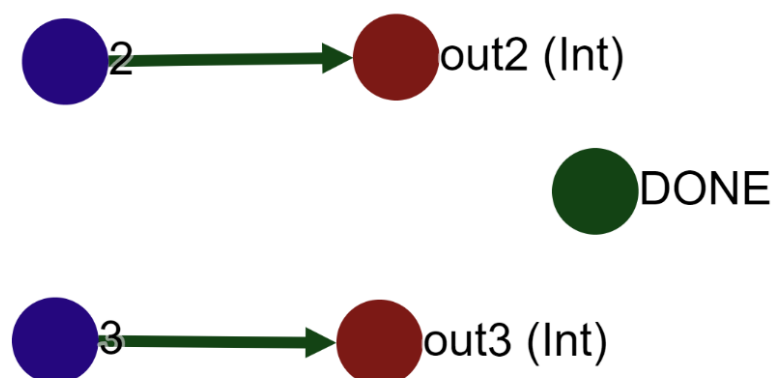


Figura 7

Condiții de salvare a funcției:

- Done nu are muchii;
- Toate nodurile au un tip („Variabila”, „Constanta” etc.);
- Dacă este funcție recursivă, trebuie să existe noduri și de intrare și de ieșire;
- Pentru fiecare funcție să fie toate porturile folosite;
- Toate muchiile trebuie să fie explicitate (se precizează ce port este folosit);
- Să existe cel puțin o variabilă ce poate să fie de ieșire;
- Să nu existe o variabilă și în lista de intrare și în cea de ieșire (nu are sens și poate produce greșeli din neatenție);

- Toate variabilele de ieșire să fie de același tip;
- Numele funcției să fie un nume valid de variabilă, asemănător limbajului C (fără spații, etc);
- Dacă funcția este recursivă, trebuie să fie un singur nod de input, tipul de date de la input și de la output să fie identic (Int de exemplu). Nodurile de tip „Recursive” trebuie să primească și să returneze același tip de date ca variabilele de intrare/ieșire.

Aceste condiții sunt destul de evidente, iar dacă nu sunt respectate se afișează o eroare specifică problemei.

O variabilă poate să fie de intrare dacă are muchii care ies. O variabilă poate să fie de ieșire dacă nu are muchii care ies și are muchii care intră.

3.3.1. Simplify

Un avantaj al folosirii Somonul-ui față de alte limbaje este dat de funcționalitatea „Simplify”. Această funcționalitate permite utilizatorului să simplifice (dacă se poate) funcția selectată sau să vadă ce alte funcții deja create se pot folosi de funcția selectată.

```

MaiMici=[],MaiMari=[];
X=funcția selectată
For( e in Funcții publice)
    If(e!=X)
    {
        Scor=Compara(X, e);
        If(Scor!=,, ,,)
            MaiMici.push(e,Scor)
        Scor=Compare(e,X);
        If(Scor!=,, ,,)
            MaiMari.push(e,Scor)
    }
SortareDupaScor(MaiMici)
SortareDupaScor(MaiMari)

```

Cod 2

Unde MaiMici reprezintă funcții mai mici care pot fi folosite în funcția X si MaiMari sunt funcții mai mari care pot folosi funcția X. Funcția de Scor folosită de mine este:

```

Scor(Mare,Mic)
{
    If(Noduri(Mare).len<Noduri(Mic).len)
        Return ,, ,;
    Ret=0
    If(Muchii(Mare).len<Muchii(Mic).len)
        Return ,, ,;
    For(e in Muchii(Mic))

```

```

{
    If(e in Muchii(Mare))
        Ret++
}
Return Ret;
}

```

Cod 3

Unde Muchii() este o funcție care returnează muchii ca perechi de date ale nodurilor (ID pentru funcții publice/private, valoarea pentru constante, tipul de date pentru variabile etc.).

Evident, se pot implementa si alte variante pentru scoruri, dar pentru ce mi-am propus a fost suficient. Ca exemplu, am făcut o funcție „testconcat” care primește o listă L, un număr X și o listă G și le concatenează în ordinea asta (funcție care urma să fie folosită la quicksort). Când am implementat funcția quicksort, aceasta părea destul de complicată încât să merite o simplificare. Evident, singura funcție recomandată a fost „testconcat”, găsind 3 muchii identice.

Chiar dacă nu returnează mereu funcții ce pot fi folosite, este utilă pentru a găsi funcții similare ce ar putea duce la idei de funcții simple si utile atât pentru funcția selectată cât și pentru cea propusă.

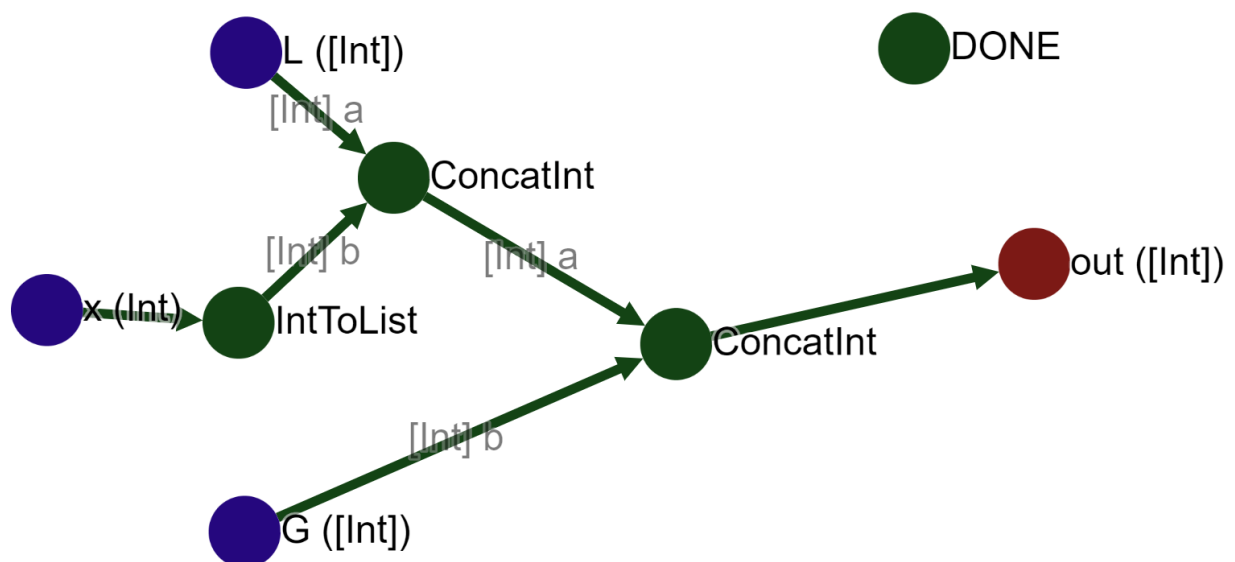


Figura 8

Smaller functions that may be used here:

```
test concats(43) : 3
```

Figura 9

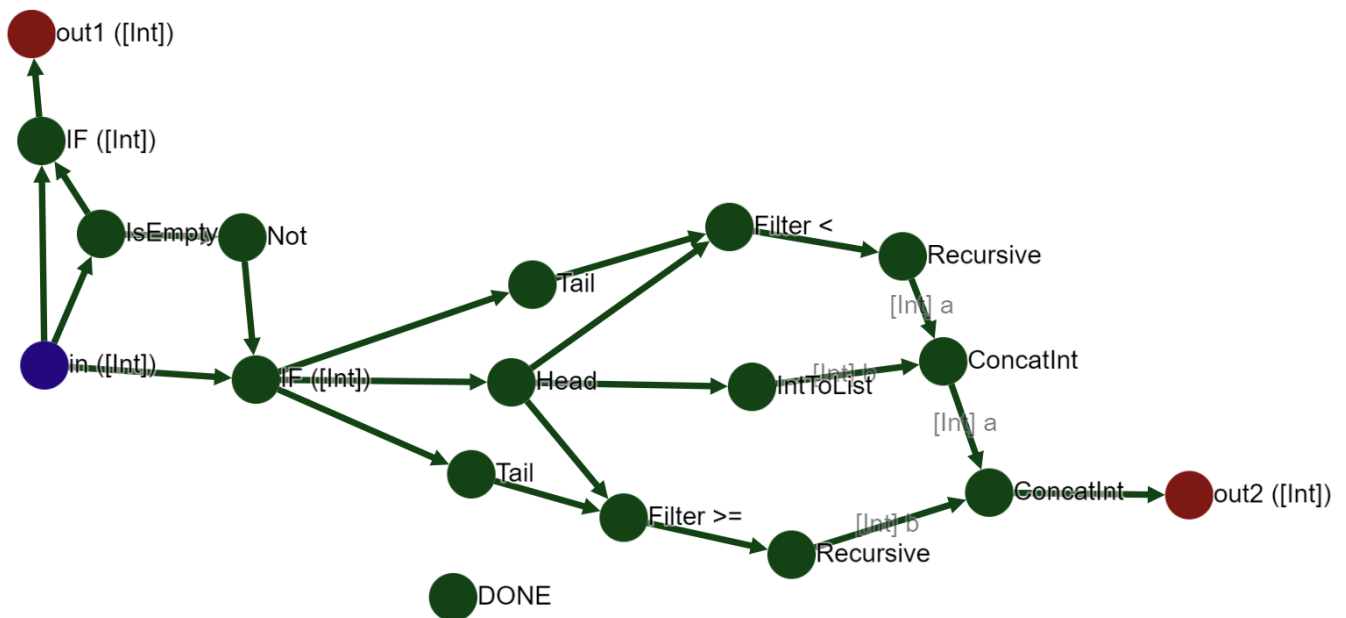


Figura 10

3.3.2. Guess function

Un alt avantaj față de alte limbaje este „Guess function” ce oferă indicații de a genera o funcție, dacă se știe tipul datelor de intrare și de ieșire. De exemplu, pentru o funcție care primește un String și un Int și returnează un Bool, îmi recomandă să transform Int-ul în String și apoi să văd dacă sunt egale iar pentru restul de 8 rezultate îmi recomandă să transform String-ul în Int și să încerc diferite funcții de comparare între numere. Un exemplu mai practic și interesant este găsirea unei funcții care primește un Mp3 și returnează o listă de imagini (partiturile asociate Mp3-ului). Există funcțiile default „Mp3ToWav”, „WavToMidi” și „MidiToSheets” (care returnează listă de imagini) deci este de așteptat să propună aceste funcții, lucru care se și întâmplă (dar recomandă mai sus, cu prioritate, funcția publică „mp3ToSheets”

care se folosește deja de acele funcții). Această funcționalitate este utilă și în a vedea dacă se poate ajunge (cu funcțiile default deja definite) de la un tip de date la altul. De exemplu, la momentul scrierii acestei lucrări, nu exista o modalitate de a ajunge de la un String la Image.

Figura 11

```
function Guesss(UnPaired,Target)
{
    Functions = Listă de funcții Default, Publice, Constructori de clasă (fiecare funcție având un
    nume, un tip (D/P/C), tipul datelor de intrare și tipul datelor de ieșire)
    var List=[];
    var How=[];
    var iteratie=1;
    var Alegeri=0;
    While
    (
    List.length!=0 //înseamnă că nu mai există soluții
    && Alegeri<9 //Maximum 9 răspunsuri
    && Iteratii / Functions.length <=10) //Maximum 10 noduri
    {
        var Head=List[0];
        if(Head.UnPaired.length==1 && Head.UnPaired[0].type==Target)
        {
```

```

        Alegeri++; //Gasit
        Afiseaza(Head.How);
    }
    List.pop();
    For(e in Functions)
    {
        Copie=Head
        Ports=e.In.Ports;
        Ok=1;
        For(port in Ports)
        {
            Gasit=0
            For(unpair in Copie.UnPaired)
            If(port.type==unpair.type)
            {
                Copie.How.Add(e,port,unpair)
                Copie.Unpaired.remove(unpair)
                Gasit=1;
                Break;
            }
            If(Gasit==0)
                Ok=0;
        }
        If(Ok)
        {
            Copie.UnPaired.Add(e.out.type)
            List.push(Copie)
        }
    }
    Iteratii++;
}
}

```

Cod 4

Pe scurt, pentru fiecare stare intermediară încerc fiecare funcție disponibilă. Dacă se potrivește o funcție (fiecare port al funcției are un tip de date de care dispune algoritmul), scot tipurile necuplate până la momentul acesta (unpaired), notez că am folosit funcția (și ce porturi) și adaug tipul de ieșire al funcției la lista de porturi ce trebuie cuplate.

Această metodă are limitarea că nu se poate folosi un nod de 2 ori (dacă vreau o funcție de la Int la Int și nu este definită o funcție de tipul „ $x \Rightarrow x+x$ ”, atunci algoritmul nu o să propună această funcție). Deci, fiecare intrare este folosită o singură dată.

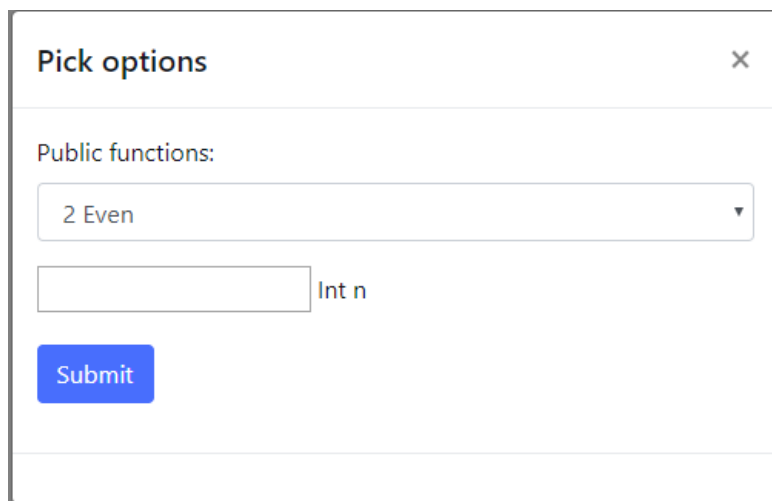
Funcțiile sunt luate în următoarea ordine: Defaults, Publice ordonate descrescător după SC, constructori de clasă ordonați descrescător după SC.

3.4. Instanțe

După ce este adăugată o funcție publică, se pot crea oricâte instanțe. Acestea sunt de fapt apeluri la funcțiile respective cu anumite valori de intrare.

3.4.1. Add instances

Întâi se alege funcția publică dorită și apoi apar porturile de intrare pentru funcția selectată. De exemplu, pentru Even există doar „Int n”. După ce este adăugată instanța (și sunt încărcate eventualele fișiere), apare un mesaj cu ID-ul instanței adăugate și un mesaj ce confirmă înregistrarea cererii.



The image shows a web interface dialog titled "Pick options" with a close button (X) in the top right corner. Inside the dialog, there is a section labeled "Public functions:" followed by a dropdown menu currently showing "2 Even". Below the dropdown is a text input field, and to its right is the label "Int n". At the bottom left of the dialog is a blue button labeled "Submit".

Figura 12

3.4.2. Automated instances

Se pot adăuga (sau șterge) funcții care se apelează la un anumit interval de timp în mod automat. Momentan, sunt implementate posibilități pentru 30 minute, o oră, o zi sau o lună. Ca o funcție să poată fi apelată automat în acest mod, ea nu trebuie să aibă porturi de intrare. De exemplu, o funcție validă este cea de Bitcoin. Chiar dacă se poate găsi un istoric pentru prețul bitcoin-ului, există date ce se pot urmări, dar care nu au un istoric (câte vizualizări are un videoclip pe Youtube, ce piese au fost pe un anumit post de radio etc.), date care pot fi folosite în diverse statistici și data mining.

3.4.3. View instances

În această pagină apar toate instanțele făcute de către utilizatorul logat cu aceste date pentru fiecare instanță:

- ID;

- Datele de intrare (având forma „Port1 = val Port2 = val ...”). În cazul în care portul are tipul de date fișier (image, video, pdf etc.), valoarea este înlocuită cu un număr ce semnifică ID-ul fișierului salvat în baza de date;

- Rezultatul funcției (în cazul în care a fost rezolvată instanța);
- Numele funcției pentru care s-a creat instanța;
- Starea (Idle = În așteptare, Taken = Există un miner care a preluat acea cerere, Fail = a apărut o eroare în rezolvarea instanței, Done = Cazul normal în care funcția a fost rezolvată cu succes);
- Momentul în care funcția a fost rezolvată (dacă a fost rezolvată sau a avut eroare);
- Cât a durat rezolvarea acelei instanțe (de exemplu 6 secunde pentru a transforma un Pdf în cele 6 imagini asociate fiecărei pagini).

Datele din tabel (din orice tabel) pot fi filtrate și exportate pentru a fi folosite de către alte programe (sau chiar de către pagina „View graphs”).

Butonul de „View end state” permite vizualizarea funcției. Pe fiecare muchie/port apare și numărul de elemente în așteptare. Această funcționalitate reprezintă principalul mod de a face „debug” (de a găsi unde sunt probleme în funcție). Ideal este să fie doar un singur port cu valoarea 1, înainte de un nod de ieșire out. Când instanța a dat eroare pentru că nu mai erau posibilități de a avansa, se pot verifica nodurile cu probleme.

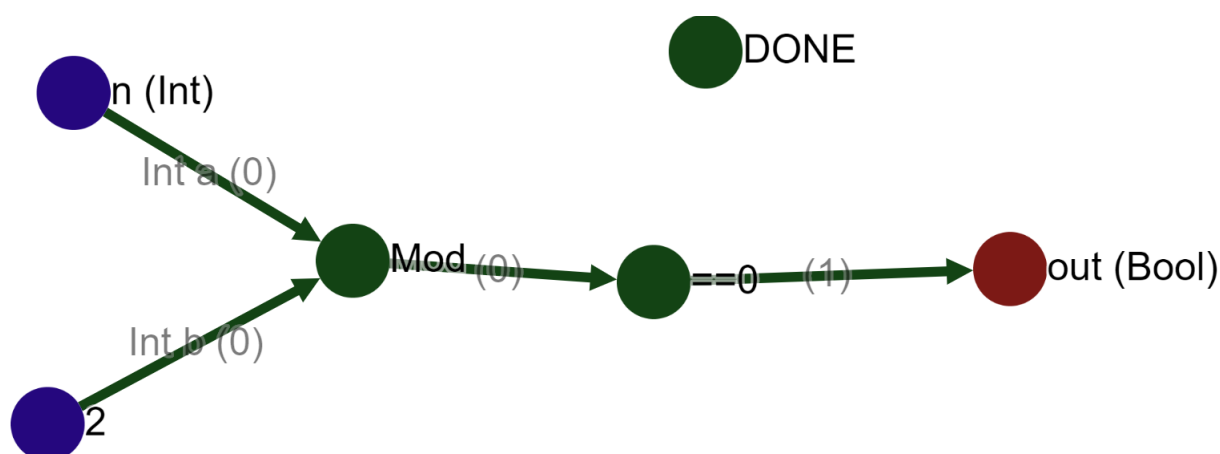


Figura 13

Următorul buton este cel de descărcare a fișierelor de ieșire (în cazul în care rezultatul este un fișier sau o listă de fișiere). Dacă trebuie descărcate mai multe fișiere (din moment ce Javascript-ul nu permite descărcarea mai multor fișiere pentru o singură acțiune (click) și din moment ce ar putea exista sute de astfel de fișiere) am decis să arhivez toate fișierele într-un zip. Există programe pe telefon ce permit deschiderea arhivelor, deci nu afectează prea mult funcționalitatea sistemului.

Ultimul buton este „Make example” și face instanța publică, vizibilă în „functions/public functions”. Butonul este colorat cu roșu deoarece datele devin (în urma acestei acțiuni) publice, deci și fișierele aferente instanței (imagini, video-uri, pdf-uri etc.). Aceste exemple îi ajută pe cei care vor să folosească funcția respectivă să își facă o idee despre cât durează o instanță (în funcție de valorile de intrare) și de modul în care arată rezultatul, fără să verifice cum arată

funcția. Sunt multe proiecte care probabil erau bune dar nu aveau un exemplu clar în care să arate cum sunt folosite acele programe și astfel au rămas nefolosite (acesta este un alt avantaj față de alte limbaje sau moduri de programare).

3.5. Clase

În pagina „Clase” se pot adăuga sau sponsoriza clase ce pot fi folosite în funcțiile publice. Acestea fiind publice nu trebuie să fie definite de fiecare dată. Pentru că pot exista mai multe clase cu același nume, acestea au și un ID (de exemplu clasa „Point” poate să fie un punct 2D, având câmpurile „Int x, Int y” sau poate să fie un punct 3D având câmpurile „Int x, Int y, Int z”). Clasele au deci un ID, un nume, câmpurile ce formează clasa respectivă, user-ul celui care a făcut clasa respectivă și SC-ul reprezentând cât a fost investit în clasă. Clasele afișate sunt sortate după SC, ca în cazul funcțiilor publice (și sunt filtrate după aceleași reguli).

Clasele au implicit doar constructori și „getteri”. Se pot realiza funcții publice care să folosească aceste clase și astfel funcțiile membre clasice sunt de fapt funcții publice care primesc ca argument de intrare o clasă.

Constructorul se comportă asemănător cu o funcție publică, atunci când primește valori pe toate porturile returnează un obiect de tipul clasei respective.

În imaginea de mai jos, a și b sunt două variabile de tip Int și Point-2 este constructorul clasei Point (cu id-ul 2). Acest constructor are nevoie de date pe porturile x și y. Am pus obiectul rezultat într-o variabilă „copie” și am folosit un getter pentru a extrage valoarea câmpului x al obiectului.

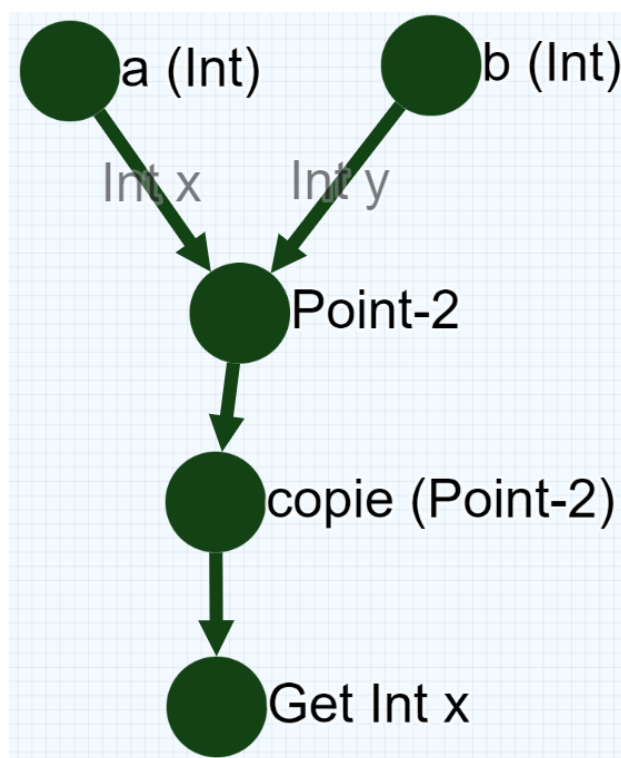


Figura 14

Ca direcții de îmbunătățire se pot adăuga: moștenirea, valori default pentru unele câmpuri, variabile din clase care să poată fi de tipul unei alte clase etc. Altă posibilitate utilă este aceea de a crea o clasă după ce a fost începută realizarea unei funcții publice. Momentan, trebuie dat refresh la pagină. Este ușor de a implementa posibilitatea de a transforma un obiect de un anumit tip (de exemplu de la Point3D, care are 3 Int-uri: x,y,z) într-un obiect mai simplu (de exemplu Point care are 2 Int-uri x și y), dacă elementele clasei mai simple au același nume și sunt de același tip.

3.6. View graphs

Datele obținute de pe site-ul <http://somon.xyz/> [16] (sau din altă parte) pot fi vizualizate pe pagina „View graphs”. Cele două tipuri de grafice suportate la acest moment sunt „Bar” și „Line”. Ambele necesită un fișier de tip CSV.

Graficul de tip bar arată o histogramă a datelor de pe prima coloană. De exemplu, pentru a testa această funcționalitate, am creat o funcție care extrage de pe un site ce melodie și ce artist cântă la un moment dat pe un post de radio. Pe lângă piese, cât de des și momentul în care au fost difuzate, am găsit și un bug pe acel site care a făcut ca timp de câteva zile să arate aceeași piesă. Pentru a salva aceste informații (și pentru a găsi acest bug) fără Somon, ar fi nevoie ca un calculator să fie pornit mereu (față de condiția de a fi un calculator deschis în toata lumea pe care să ruleze Somon-ul la momentul respectiv).

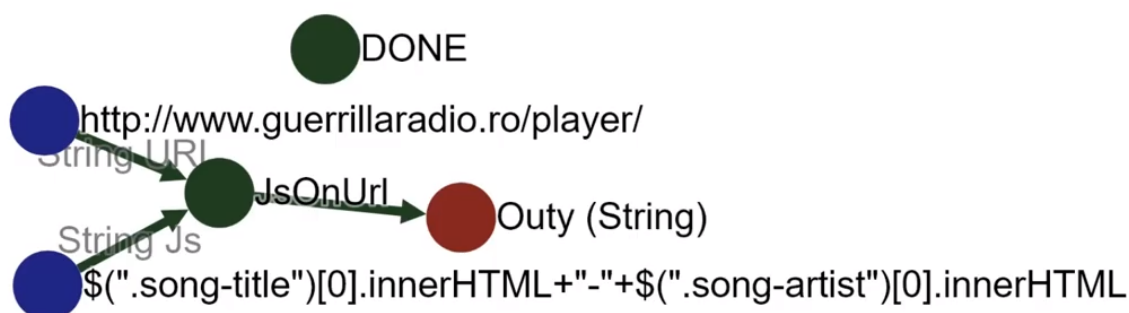


Figura 15

Instances
Functions
Download
View graphs

ID	InVar	OutValue	Name	State	TimestampDone	Duration (ms)
739		Feeling Good-MUSE	Guer	Done	2018-08-20 12:30:19	6066
737		Feeling Good-MUSE	Guer	Done	2018-08-20 12:00:17	5127
736		Feeling Good-MUSE	Guer	Done	2018-08-20 11:30:19	2899
735		Feeling Good-MUSE	Guer	Done	2018-08-20 11:00:38	5079
733		Feeling Good-MUSE	Guer	Done	2018-08-20 10:30:20	4930
731		Feeling Good-MUSE	Guer	Done	2018-08-20 10:00:19	9492
730		two nights (Radio Edit)-LYKKE LI	Guer	Done	2018-08-20 09:30:16	2952
729		Radio Guerrilla-	Guer	Done	2018-08-20 09:00:36	6024
727		Happier-MARSHMELLO & BASTILLE	Guer	Done	2018-08-20 08:30:19	7008
726		Afterglow-INXS	Guer	Done	2018-08-20 08:00:35	2746
724		Jumpsuit-TWENTY ONE PILOTS	Guer	Done	2018-08-20 07:30:15	6576
723		Rebel Yell-BILLY IDOL	Guer	Done	2018-08-20 07:00:40	5687
721		Shed a Tear (Radio Edit)-KODALINE	Guer	Done	2018-08-20 06:30:07	2757
720		Numb-LINKIN PARK	Guer	Done	2018-08-20 06:00:38	3581
718		Radio Guerrilla-	Guer	Done	2018-08-20 05:30:06	3232
717		Let Me Go-HAIM	Guer	Done	2018-08-20 05:00:39	3452
715		Heavy, California-JUNGLE	Guer	Done	2018-08-20 04:30:09	3409
713		Radio Guerrilla-	Guer	Done	2018-08-20 04:00:10	3340
712		Bohemian Rhapsody-QUEEN	Guer	Done	2018-08-20 03:30:14	3372
710		Radio Guerrilla-	Guer	Done	2018-08-20 03:00:15	3308
709		Goodnight Irene-KEITH RICHARDS	Guer	Done	2018-08-20 02:30:19	3416
707		Radio Guerrilla-	Guer	Done	2018-08-20 02:00:19	3404
706		Walking In The Air-NIGHTWISH	Guer	Done	2018-08-20 01:30:08	3460

Search:
Copy
Excel
CSV
PDF
Showing 1 to 297 of 297 entries (filtered from 612 total entries)

Figura 16

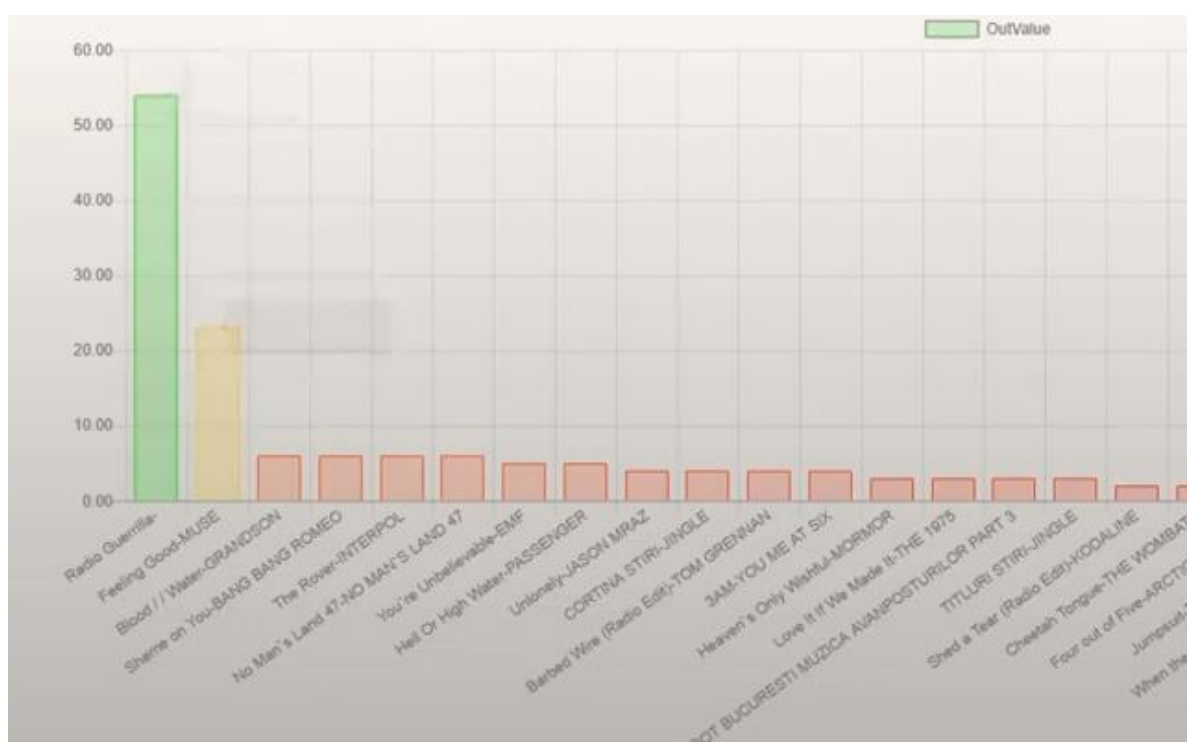


Figura 17

Al doilea tip de grafic este „Line”. Acesta necesită ca prima coloană din CSV să fie o dată (zi, lună și an) sau dată și timp (zi, lună, an, oră, minut, secundă). Apoi pot fi folosite oricâte coloane care să aibă valori numerice. Primul rând din fișier reprezintă numele ce urmează să fie folosite pentru a arăta liniile respective. De exemplu, se pot descărca datele despre cursul Leu-Euro [13]. Chiar dacă există deja grafice pe internet despre cursurile valutare, unele date trebuie salvate pentru că nu există un istoric al lor. De exemplu, nu există un istoric pentru rank-ul într-

un joc online sau pentru numărul de semnături strânse de către partide. Folosind vizualizarea datelor se pot estima valori viitoare sau dacă au apărut evenimente care nu au sens (să scadă numărul de semnături afișat pe site, din moment ce acestea trebuie doar să crească, lucru pe care l-am întâlnit).

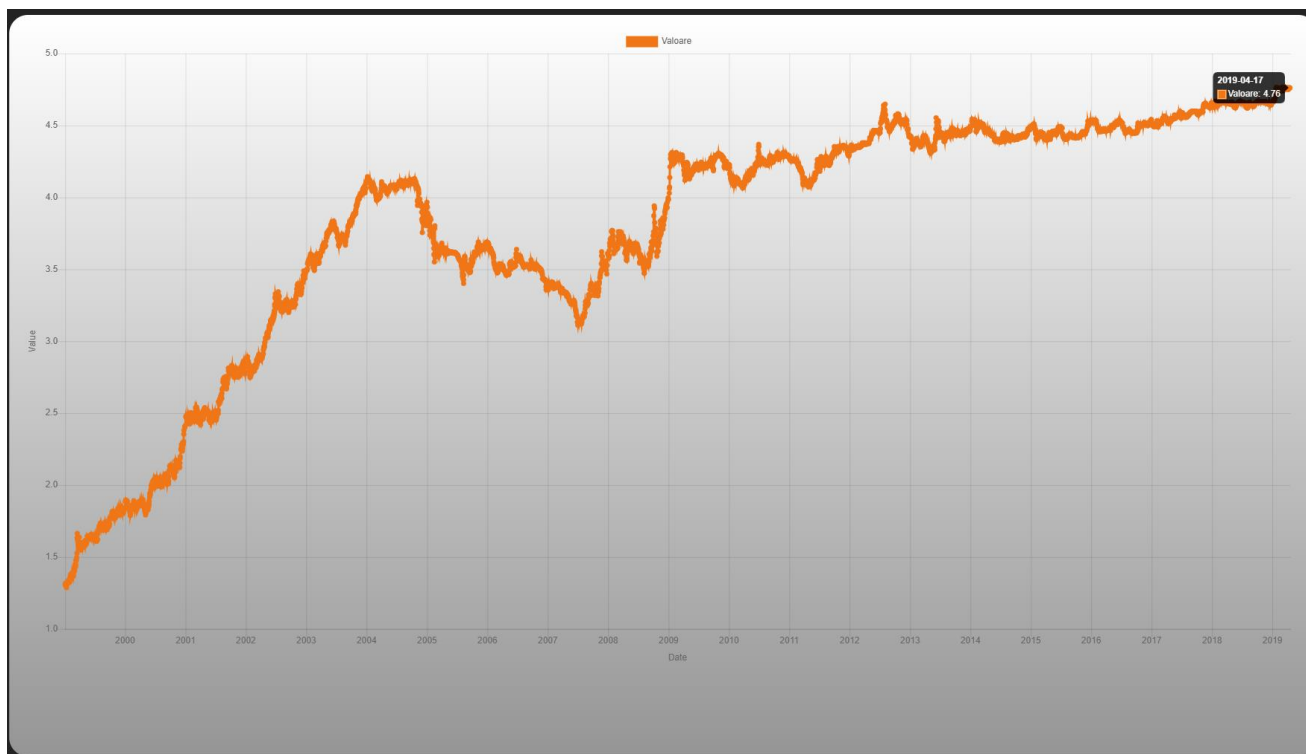


Figura 18

Evident se pot adăuga mai multe tipuri de grafice (candle, scatter etc.) dar acestea două sunt suficiente pentru a arăta un concept.

3.7. Users

În pagina „users” se pot vedea toți utilizatorii care și-au creat un cont. ID-ul este unic pentru fiecare cont în sistem, username-ul este folosit în alte module (de exemplu pentru a filtra funcțiile publice făcute de către un anumit utilizator) și câteva informații despre acel utilizator precum: numărul de funcții făcute, instanțe create, instanțe rezolvate, timpul computațional utilizat rezolvând instanțe. Se mai pot adăuga cu ușurință diverse alte măsurători.

3.8. Functions statistics

În această pagină se pot vedea funcțiile care au fost folosite în timpul minării, atât cele publice cât și cele default (o altă diferență față de alte limbaje). Această valoare poate să fie un indicator în alegerea (sau încrederea acordată) unei funcții publice. De exemplu, funcția publică IsEmpty (cu ID-ul 54) a fost apelată de 39434 ori.

Type	Name	Nr
Default	Mod (3)	10220950
Default	==0 (7)	10220949
Default	.. (27)	39493
Default	==[Int] (39)	39476
Default	Div (16)	39454
Public	IsEmpty(54)	39434

Figura 19

Se poate folosi în viitor pentru a sugera funcții. De exemplu, când ai un nod cu anumite restricții de intrare/ieșire primești un Int și returnezi un String. Pe lângă SC, un alt factor în a sorta funcțiile disponibile poate să fie numărul de apelări. Se pot memora și perechi de funcții apelate consecutiv (funcția default Mod are aproximativ tot atâtea apelări ca funcția „==0”). Aceasta este o altă informație dinamică, ce poate ajuta utilizatorul să fie mai eficient.

3.9. Settings

Deoarece pot exista multe funcții sau clase de test, sau funcții care se pot bloca (din cauza IF-ului) și trebuie reparate, acestea ar putea încărca prea mult opțiunile utilizatorului de a alege o funcție pe care să o folosească. Utilizatorul are opțiunea să vizualizeze funcții de la oricare utilizator sau doar făcute de către el, ambele cazuri având un prag minim de SC pentru a fi luate în considerare. Cum funcțiile au inițial 0 SC, dacă există un prag diferit de 0 în condiția de mai sus, nu ar apărea o funcție nou creată. De aceea, o altă opțiune este de a vedea funcțiile făcute în ultimele x ore.

Legat de instanțe, se poate seta cine poate rezolva instanțele create: toți utilizatorii sau doar utilizatorul care a adăugat instanța. La fel, se poate seta și pentru cine rezolvă instanțe: doar pentru el sau și pentru ceilalți. Această setare există deoarece fișierele folosite într-o instanță pot ajunge pe calculatoarele altor utilizatori, ceea ce nu este mereu de dorit. A doua parte a setării previne descărcarea și deschiderea fișierelor necunoscute de la alți utilizatori.

Sunt multe alte idei de setări care nu au ajuns să fie implementate. De exemplu, să poți alege utilizatori cărora să le rezolvi instanțe, să poți alege ce funcții default sunt folosite când rezolvi instanțe pentru alții (nu prea contează pentru +, -, Mod, ==0 etc. , dar când sunt descărcate fișiere și accesate site-uri necunoscute e importantă alegerea).

3.10. Download

Cele două link-uri din Download permit descărcarea miner-ului și fișierului de configurare pentru utilizatorul care accesează link-ul. Fișierul de configurare (config.uap) trebuie pus în folder-ul Miner pentru a îi „spune” miner-ului ce utilizator rezolvă instanțele (deci cine primește SC pentru rezolvarea acestora).

4. Miner

Miner-ul este o aplicație scrisă în C++ care se conectează la o bază de date, caută o instanță liberă, o rezolvă folosind eventual programe ajutătoare și returnează rezultatul către baza de date.

Mai detaliat, după ce se verifică fișierul de configurare (ce conține date despre cel care a descărcat acel fișier) se verifică ce fel de instanțe dorește să rezolve utilizatorul respectiv (pentru toți sau doar pentru el). Începe apoi căutarea unor astfel de instanțe care sunt Idle (în așteptare) la fiecare câteva secunde. După găsire, este marcată găsirea unei instanțe (prin schimbarea stării din Idle în Taken), sunt încărcate definițiile de clase și de funcții publice (deoarece este posibil să fi apărut unele noi) de la ultima instanță rezolvată. Este pornit un cronometru ce măsoară timpul folosit pentru rezolvarea problemei (în funcție de care se calculează SC-ul câștigat la sfârșit). Se rezolvă funcția, cronometrul este oprit (deoarece restul acțiunilor nu țin neapărat de rezolvare), se salvează numărul elementelor în așteptare de pe fiecare port, se schimbă informațiile despre instanță, adică: starea devine Done sau Fail, în OutValue se pune rezultatul funcției sau un mesaj ajutător în cazul în care a fost Fail, cât timp a durat rezolvarea instanței. Este crescut SC-ul utilizatorului (la un minut lucrat, se primește un SC). Se adaugă informațiile pentru statistici (de câte ori a fost folosită fiecare funcție în rezolvare, atât cele publice cât și cele default).

Rezolvarea unei instanțe:

- se golesc porturile (de la o eventuală instanță rezolvată înainte);
- datele de intrare sunt parsate și puse pe porturile variabilelor de intrare ale funcției;
- se intră în loop-ul principal care verifică dacă este rezolvată instanța (dacă există o variabilă de ieșire și are un rezultat pe un port) și dacă nu, verifică ce noduri pot avansa în acel moment.

Pentru „avansarea” unui nod, trebuie să ne uităm la tipul acestuia. Toate tipurile verifică întâi dacă au cel puțin o valoare pe fiecare port de intrare (în afară de constante, care nu au porturi de intrare). De asemenea, toate tipurile nodurilor trimit (în afară de IF care poate trimite sau nu) rezultatul acelui nod pe toate muchiile de ieșire, către alte noduri (deci output-ul unui nod poate fi trimis către oricâte noduri).

4.1. Default

Funcțiile default folosite momentan în aplicație sunt următoarele (cu roșu sunt trecute funcțiile de bază, cu verde funcțiile care transformă fișiere și cu albastru funcții legate de IA):

ID	Name	InVar	OutType
1	+	Int a,Int b	Int
2	-	Int a,Int b	Int
3	Mod	Int a,Int b	Int
4	--	Int a	Int
5	++	Int a	Int
7	==0	Int a	Bool
8	or	Bool a,Bool b	Bool
9	and	Bool a,Bool b	Bool
11	ifte	Bool cond,Int then,Int else	Int
12	>	Int a,Int b	Bool
13	<	Int a,Int b	Bool
14	WavToMp3	Wav in	Mp3
15	==Int	Int a,Int b	Bool
16	Div	Int a,Int b	Int
17	Int2Str	Int a	String
18	Str2Int	String a	Int
19	Reverse	String a	String
20	==Str	String a,String b	Bool
21	*	Int a,Int b	Int
22	JsOnUrl	String URL,String Js	String
23	Head	[Int] List	Int
24	GetPalette	Image img,Int colors	Image
25	ScaleImg	Image img,Int percent	Image
26	ScaleVid	Video vid,Int percent	Video
27	..	Int a,Int b	[Int]
28	ReverseGif	Gif gif	Gif
29	Not	Bool a	Bool
30	!=Int	Int a,Int b	Bool
31	ConcatInt	[Int] a,[Int] b	[Int]
32	DateTimeNow		String
33	Mp3ToWav	Mp3 in	Wav
34	WavToMidi	Wav in	Mid
35	MidiToSheets	Mid in	[Image]
36	AudioAndImageToVideo	Image img,Mp3 mp3	Video
37	StringToWav	String in	Wav
38	IntToList	Int a	[Int]
39	==[Int]	[Int] a,[Int] b	Bool
41	Tail	[Int] List	[Int]
42	OCR	Image img	String
43	PdfToImages	Pdf in	[Image]
44	TrainSVM	CSV in	SVM
45	PredictSVM	CSV in,SVM trained	Txt

46	StringKernel	Zip in	Txt
47	WsdTLDR	String sentence,String word	String
48	WsdDetails	String sentence,String word	Txt
49	ImageFeatures	Image img	[Double]
50	MachineLearning	String Alg,String Norm,CSV Train,CSV Test	Txt
51	EverythingSolver	CSV Train,Int sec	Txt

Tabel 1

Pentru nodurile Default, cele mai vechi valori de pe fiecare port de intrare sunt extrase și puse într-un obiect ce este trimis unei funcții (SolveDefault). Această funcție primește un număr (ce definește funcția, ID-ul din baza de date) și obiectul menționat anterior (ce poate fi considerat un map de la string la string). De exemplu, pentru funcția „+”, partea de cod relevantă este:

```
if(Functie=="1")///plus
    return ss(InfInt(data.Get("Int a"))+InfInt(data.Get("Int b")));
```

Cod 5

Pe scurt, „ss” transformă orice în string, „InfInt” transformă un string în int iar „data.Get(port)” returnează valoarea stringul-ui de pe portul respectiv.

Un alt exemplu de funcție default (ce folosește și fișiere) este cea cu ID-ul 36, „AudioAndImageToVideo”:

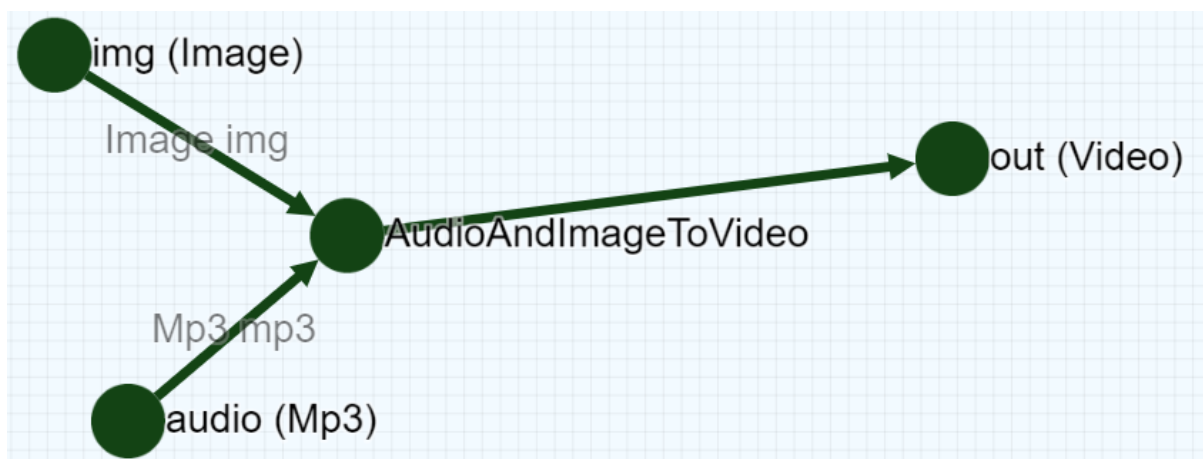


Figura 20

Am verificat faptul că nu apare instanța respectivă în istoricul celui care a minat, dacă este altul decât cel care a creat instanța (dar a avut acces la fișiere în timpul rezolvării instanței).

```

if(Functie=="36")///AudioAndImageToVideo
{
    Linie LImg=SaveFile(data.Get("Image img"));
    Linie LMp3=SaveFile(data.Get("Mp3 mp3"));

    exec(ss("Exes\\ffmpeg.exe -loop 1 -i \"res/\"+LImg["FileName"]+"\" -i \"res/\"+LMp3["FileName"]+"\" -vf
\"scale=ceil(iw*0.5)*2:-2\" -c:a copy -c:v libx264 -shortest res/out.mp4 ").c_str());

    string ID=ss(QueryToCall("INSERT INTO BlobFiles(BlobFile, FileName,UserID) VALUES
("+FileToDB("res/out.mp4")+",'out.mp4','+InstanceStarterID+"))").nLastInsertId);
    DeleteDir("res");
    return ID;
}

```

Cod 6

În această bucată de cod, cele două fișiere de intrare sunt salvate local, în folder-ul minerului. Folosind programul „ffmpeg” (care știe să transforme o imagine și un mp3 într-un mp4) este creat fișierul dorit, care este apoi uploadat pe server (și se returnează ID-ul fișierului).

4.2. Public

În mod similar se rezolvă și nodurile de tip Public. Porturile și valorile lor sunt concatenate într-un string iar apoi este apelată (recursiv) funcția care rezolvă instanțe.

4.3. Variabile

Variabilele au un singur port de intrare. Dacă au o valoare pe acel port, aceasta este scoasă din coadă și trimisă către toate celelalte noduri destinație. Inițial, când nu existau constante, variabilele erau folosite pe post de constante, nodurile având bucle. Încă se poate folosi în acest mod pentru a transforma un rezultat al unei funcții într-o constantă iar funcția este apelată doar o dată. De exemplu, în cazul în care este variabil rezultatul funcției respective (Bitcoin) și se dorește folosirea acelei informații de mai multe ori.

4.4. Constant

Constantele trimit valoarea definită la crearea funcției către nodurile destinație doar dacă pe acele porturi nu se află deja date. Acest lucru ajută la oprirea instanței (cu Fail) în momentul în care nici un nod nu a mai avansat. Dacă nu ar fi această verificare a destinației și funcția nu ar avea șanse reale să se termine (să fie un nod IF care verifică dacă o constantă este egală cu o altă constantă diferită), ar părea că încă se mai întâmplă ceva în acea instanță dacă tot aș adăuga date. Coada ar putea deveni prea mare și ar fi o problemă cu memoria și timpul de execuție ar fi irosit generând valori de care nu este nevoie în momentul respectiv (coada având cel puțin o valoare în ea).

4.5. IF

Acest nod permite sau nu trecerea datelor. Are porturile „Bool cond” ce reprezintă condiția care trebuie verificată și portul „Tip a” ce reprezintă valoarea care va trece sau nu de acest nod. „Tip” poate lua valorile tipurilor obișnuite de date (Int, String, Pdf etc.), acesta fiind definit în momentul în care a fost creată funcția publică.

4.6. Class

Acest nod creează un obiect (un tip de date) al clasei asociate nodului (clasa a fost aleasă când a fost definită funcția). Dacă toate câmpurile au cel puțin o valoare, datele se concatenează într-un singur șir de caractere, iar acest șir este trimis mai departe ca obiect. Aceste obiecte se pot accesa folosind nodul de tip „ClassGetter”.

4.7. ClassGetter

Are un singur port de intrare (portul gol, asemănător cu cel al variabilelor) și pe acest port așteaptă obiecte de un anumit tip. Când se primește o valoare, obiectul (care este în realitate un șir de caractere) este parsat și apoi este extras câmpul cerut.

4.8. Recursive

Nodul Recursive are cea mai mică prioritate. Acesta are un singur port de intrare, cel gol la fel ca la variabile. Seamănă foarte mult cu nodul Public, însă în momentul definirii funcției ce conține un nod Recursive nu există definite clar datele de intrare și cele de ieșire, ci doar promisiuni că o să aibă antetul respectiv.

5. Baza de date

AutomatedInstances	Function,UserID,Type,ID
BlobFiles	FileName,BlobFile,ID,UserID,Added
Classes	TimestampAdd,InVar,SC,UserID,Name,ID
DefaultFunctions	Name,InVar,OutType,SomonCode,ID
Examples	InstanceID,PublicID
FunctionUsage	ID,Nr,Type
Instances	Starter,EdgeSizes,Worker,InVar,State,F,OutValue,ForUser,ID,Duration,TimestampAdd,TimestampDone
PublicFunctions	UserID,Dependency,JsonString,OutType,OutVar,ID,Name,Description,TimestampAdd,SC,InVar
Requests	Name,Description,Type,User,SC,State,ID
RequestsSolutions	FunctionID,RequestID,SC
Users	Email,TruncateInstance,AddInstancesFor,SolveInstancesFor,ShowMineHours,ShowSC,ShowAllMine,ID,SC,Username>Password,JoinTimestamp
VariableTypes	Name,ID

Figura 21

Pe scurt despre fiecare tabelă:

AutomatedInstances. Type-ul este un enum ('30 min', '1 hour', '1 day', '1 month') care arată cât de des trebuie adăugată instanța. UserID-ul este ID-ul userului care poate șterge acea intrare (și cel care poate să vadă pe site). Function este ID-ul funcției publice fără porturi de intrare care este apelată.

În **BlobFiles** sunt păstrate fișierele folosite în diferite instanțe. Numele și extensia sunt memorate în **FileName**. Pe cât posibil, se păstrează numele fișierului (când este transformat dintr-un tip în altul, precum .mp3 către .wav). UserID este ID-ul utilizatorului care a făcut instanța. Fișierele se pot descărca de pe site doar de către utilizatorul care a făcut instanța (dar se pot descărca și de către alți utilizatori dacă a fost permis în setări rezolvarea de către alte persoane). **Added** arată când a fost adăugat un anumit fișier (nu este esențial pentru sistem).

Classes conține definițiile pentru clasele create. Numele, utilizatorul care a creat clasa, cât SC a fost investit și câmpurile clasei (ținute în coloana **InVar**, numită așa deoarece câmpurile unei clase seamănă foarte mult cu porturile de intrare ale funcțiilor). Exemplu de **InVar** pentru o clasa **Point** este „Int x,Int y”.

În **DefaultFunctions** stau funcțiile implicite, definite în miner (miner care caută ID-ul din această tabelă pentru a identifica funcția folosită). **Name** este numele funcției, în **InVar** sunt porturile de intrare (de exemplu pentru **JsOnUrl**, **InVar** este „String URL,String Js”). **OutType** este tipul de date rezultat (tot pentru **JsOnUrl** avem „String”). **SomonCode** este o informație opțională ce arată echivalentul funcției în C/C++ (pentru funcția **Mod**, **SomonCode**-ul asociat este „r=a%b”).

Tabela **Examples** este o tabelă asociativă, aceasta conține ID-ul funcției publice și ID-ul instanței.

FunctionUsage este folosit doar pentru statistici, numără de câte ori au fost apelate funcțiile (**Type** poate să fie **Default** sau **Public**). Coloana **Nr** reprezintă numărul de apeluri pe care l-a avut funcția respectivă.

Instances este tabelul cu instanțele făcute. În **InVar** sunt valorile de intrare (**Port1=Val Port2=Val ...**). **OutValue** este rezultatul instanței după ce este rezolvată, **F** este funcția publică asociată instanței, **Starter** este utilizatorul care a inițiat instanța, **Worker** este cel care a rezolvat

funcția, ForUser este fie egal cu Starter (dacă în setări acel utilizator a ales ca doar el să poată rezolva instanțele lui) fie 0 (caz în care oricine le poate rezolva). Duration este durata de calcul (în milisecunde). TimestampAdd și TimestampDone sunt pentru statistici. EdgeSizes reprezintă dimensiunea cozilor la sfârșitul instanței (folosit pentru butonul „View end state”).

PublicFunctions: ID-ul funcției, numele și descrierea funcției, data la care a fost adăugată funcția, user-ul care a creat funcția, câți SC au fost investiți, variabilele de intrare și de ieșire separate prin virgulă (de exemplu pentru „>=” avem InVar= „Int a,Int b” OutVar=„Bool out”), tipul de date al rezultatului, ID-ul funcțiilor default (dependințele) și un text Json ce memorează datele asociate funcției (toate nodurile, poziția lor, muchiile dintre noduri, culoarea lor etc.).

Requests: tipul requestului (Public = pentru cazul în care se poate face o funcție folosind funcțiile existente sau Default în caz contrar), numele și descrierea cererii, ID-ul utilizatorului care a creat cererea, câți SC au fost investiți în request și starea curentă (Active/Done).

RequestsSolutions: ID-ul requestului inițial, ID-ul funcției publice propuse ca soluție, câți SC au fost investiți în acea soluție.

Users: ID-ul clientului, email-ul, parola, username-ul, câți SC are, când a fost creat contul, dacă vrea să vadă toate funcțiile publice sau doar pe cele făcute de către el, pragul de SC pentru apariția funcțiilor publice în lista lui, ShowMineHours pentru a vedea funcțiile publice făcute recent (care inițial au 0 SC) dar nu au încă destui SC pentru a trece peste pragul menționat anterior, dacă vrea să rezolve instanțe și pentru alți utilizatori sau doar pentru el, dacă instanțele adăugate de el să poată fi rezolvate și de către alți utilizatori sau doar de către el și dacă output-ul să fie trunchiat sau nu dacă depășește 50 de caractere.

VariableTypes: Name este tipul de date (Int, String etc) și un ID.

6. Exemple

6.1. Exemple cu funcții existente

6.1.1. Basic

Majoritatea funcțiilor au nevoie de aceste funcții de bază (atât cele simple, cât și cele complicate). De exemplu: dacă un număr este par sau nu, sortarea unei liste de numere, dacă o listă este goală sau nu etc.

6.1.2. File transform

Posibilitatea de a crea fișiere, de a le modifica și de a extrage informații din ele este foarte importantă în multe domenii. De exemplu, pentru inteligență artificială este importantă redimensionarea imaginilor pentru probleme de vision, prelucrarea textelor pentru probleme de NLP și extragerea feature-urilor din fișiere pentru machine learning.

Am avut nevoie să fac un Gif să meargă invers (reverse, să pornească de la sfârșit), de aceea am adăugat această funcție în Somon (ReverseGif, funcția default 28), iar dacă o să am nevoie altă dată să aplic și alte efecte, o să fie mult mai ușor să le combin.

Cel mai bun exemplu de „flow” momentan este dat de funcția publică „mp3ToSheets”. Sunt folosite funcțiile Mp3ToWav, WavToMidi, MidiToSheets pentru a transforma un fișier în partituri. Nu există foarte multe programe care să facă acest lucru și majoritatea costă bani. Această înșiruire poate să fie continuată și, în loc de a porni de la un mp3, se poate porni de la un link de Youtube sau numele unei piese care apoi să fie căutată pe Youtube (există și piese fără drepturi de autor).

Am adăugat funcția „AudioAndImageToVideo” după ce mi-am amintit că a avut cineva nevoie de asta. Mi-a fost cerut să fac un program care încarcă pe Youtube piese:

“Hi there!

I have a very big music collection that i want to upload to youtube for history purposes. Its actually allowed if you dont advertize those videos etc.

So what i would need is a program that lets me put X files in queue and then uploads the mp3 as a video with just the name of the file displaying in the video section.

So i can basically drop all those files in and the video creation and everything else happens in the background. Is that anywhere near possible?

If not with java.. which programming language would be? and if the task is impossible that way... what would be possible that comes close to those requirements?

As i am a poor student my budget is 60\$ max unfortunately.

Cheers”

Jumătate de problemă constă în transformarea fișierelor mp3 în videouri, prin adăugarea unei imagini de fundal, ceea ce se poate face acum în Somon. Din moment ce am rezolvat atunci

problema în C++, se poate adăuga și partea de upload către Youtube sub forma unui executabil. Astfel, problema se putea rezolva simplu, cu două funcții default. Funcția publică nou creată poate să fie folosită apoi de către oricine și oricine putea să rezolve problema. Asta a fost acum 2 ani și între timp am pierdut codul respectiv, iar acum dacă trebuie să fac din nou acel proiect, trebuie să caut iar o librărie ce permite uploadarea de fișiere pe Youtube. Puteam foarte bine să adaug executabilul la Somon (dacă acesta exista), să notez parametrii de intrare și tipul rezultatului, iar soluția rămânea astfel salvată.

6.1.3. Inteligența artificială

6.1.3.1. Vision

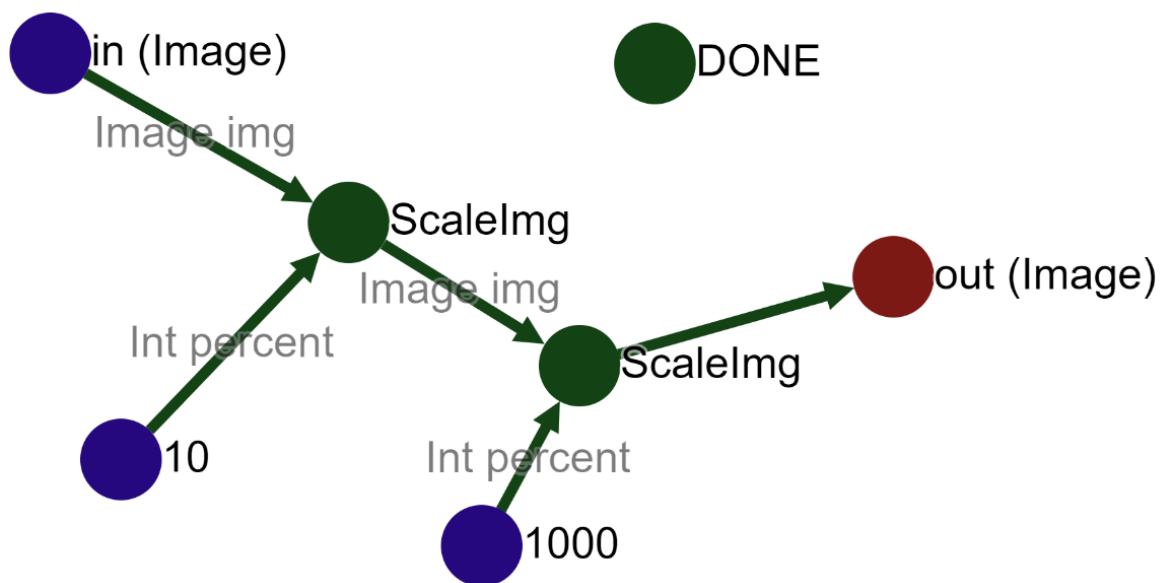


Figura 22

Funcția de mai sus poate genera imagini blurate pentru a folosi aceste date la antrenare (pentru un rezultat mai bun și pe imagini cu rezoluție mai slabă sau mișcate). Poate să fie folosită și pentru a genera date pentru un program care face imagini mai clare.

Obținerea de feature-uri este esențială pentru algoritmi de machine learning. Pentru imagini am adăugat funcția default „ImageFeatures” care primește o imagine și întoarce 8 numere reale (Double). Cele 8 numere reprezintă procentul de culori care se încadrează în 8 bin / buckets[17]. Cele 8 bin-uri în ordine sunt:

$(0,0,0) \Rightarrow (128,128,128)$
 $(0,0,128) \Rightarrow (128,128,255)$
 $(0,128,0) \Rightarrow (128,255,128)$
 $(0,128,128) \Rightarrow (128,255,255)$
 $(128,0,0) \Rightarrow (255,128,128)$

(128,0,128) => (255,128,255)
(128,128,0) => (255,255,128)
(128,128,128) => (255,255,255)

De exemplu, pentru imaginea de mai jos se obține lista: [0.135, 0.229, 0.162, 0.374, 0.000, 0.000, 0.022, 0.079]. Cum primele 4 numere au asociate bin-uri care au valoare mică pentru canalul roșu și în imagine nu apare această culoare (sau culori asemănătoare precum galben, mov etc.), 90% dintre date sunt exprimate de către bin-urile semnificative pentru albastru și verde. Ultimul bin are aproape 8% deoarece este asociat culorilor deschise (aproape alb). Evident că pot fi folosite oricâte bin-uri și eventual construite altfel. Funcția poate să fie folosită pentru a grupa imagini asemănătoare (indiferent de dimensiunea lor) sau pentru a detecta dacă o imagine este alb-negru sau nu etc. Imaginile alb-negru au valori doar în primul și ultimul bin. Pentru aceeași imagine dar transformată în alb-negru, se obțin valorile: [0.692, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.308].



Figura 23 – Sursa [32]

```
V=np.zeros(2*2*2)
import sys
img =Image.open( sys.argv[1] )
pixels = img.load() # create the pixel map
for i in range(img.size[0]): # for every col:
    for j in range(img.size[1]): # For every row
        r,g,b=img.getpixel((i,j))
        r=int(r/128)
        g=int(g/128)
        b=int(b/128)
        V[r*2*2+g*2+b]+=1
```

```

V/=sum(V)
for idx,v in enumerate(V):
    if(idx==0):
        print("%3.3f" % v,end="")
    else:
        print(", %3.3f" % v,end="")

```

Cod 7

Pot exista și alte funcții asemănătoare și rezultatul lor să fie concatenat. Funcțiile acestea care extrag feature-uri din imagini pot fi găsite imediat deoarece funcțiile pot fi filtrate după tipul datelor de intrare și tipul datelor de ieșire. Momentan, există o singură funcție default care primește o imagine și returnează o listă de Double.

Alte două funcții legate de vision (ce pot da feature-uri din imagini) sunt OCR (care citește textul din imagine) și GetPalette (care arată cele mai folosite culori).

6.1.3.2. Text - String kernel

Un program foarte interesant, la care am lucrat la master, folosea String Kernel. Această funcție folosește date de antrenare (fișiere text) clasificate într-un anumit mod. Primind datele de antrenare și de test, ea generează o matrice de similarități între fișiere. După ce se antrenează pe datele respective, propune clase pentru fișierele de test. În lucrarea unde este prezentată această metodă[18] sunt prezentate 3 funcții de similaritate între texte folosind numărul de apariții ale n-gramelor în textele respective. Când se compară două fișiere, similaritatea se obține adunând rezultatele unei funcții de similaritate pentru fiecare n-gramă care se găsește în ambele fișiere:

- **Presence** returnează 1;
- **Spectrum** întoarce produsul dintre numărul de apariții a n-gramei în primul fișier și numărul de apariții a n-gramei în al doilea fișier;
- **Intersect** calculează minimum dintre numărul de apariții a n-gramei în primul fișier și numărul de apariții a n-gramei în al doilea fișier.

Acest program este generic și a fost aplicat în multe competiții diferite și a obținut rezultate bune. De exemplu, poate să fie folosit pentru a determina dacă un text este scris de un om sau de către un robot, de un bărbat sau o femeie, dacă un text în engleză este scris de către un britanic sau un american, care este țara de origine a celui care a scris un text în engleză, dacă este un text pozitiv sau negativ etc. Frumusețea programului este aceea că poate să fie folosit în orice limbă, deoarece se lucrează la nivel de caractere.

Când am folosit pentru un concurs acest algoritm am avut rezultate bune. Tot căutând metode de a îmbunătăți rezultatul, am găsit o nouă funcție de similaritate ce creștea acuratețea programului semnificativ (o funcție care pentru o n-gramă și cele 2 numere ce semnifică numărul de apariții ale n-gramei în cele 2 fișiere, întoarce minimul dintre cele două numere supra maximul celor două numere). Există și o implementare în Java pe internet[19]. Dacă era scris

programul respectiv în Somon, era mult mai ușor de schimbat funcția de similaritate și nu era nevoie de un compilator/editor de Java pentru a fi rulat/modificat programul.

6.1.3.3. Text - WSD

Un alt proiect interesant de la master legat de text a fost WSD (Word Sense Disambiguation) [20; 21; 22]. În acest proiect, se alege sensul cel mai probabil al unui cuvânt într-un anumit context. Programul implementat de mine funcționează deocamdată doar pentru limba engleză și este bazat pe cunoștințe (precum definiții pentru fiecare cuvânt și legături între cuvinte). Programul caută pentru fiecare sens posibil cuvinte din context care sunt identice sau similare. Se compară definiția sensului (cu eventuale exemple) cu definițiile sensurilor cuvintelor din context, unde contextul reprezintă câteva cuvinte înainte și după cuvântul căutat.

De exemplu:

- We cannot fill more gasoline in the tank. (rezervor de mașină)

Possible synsets

1 an enclosed armored military vehicle; has a cannon and moves on caterpillar treads

2 a large (usually metallic) vessel for holding gases or liquids

3 as much as a tank will hold

4 a freight car that transports liquids or gases in bulk

5 a cell for violent prisoners

Prediction:

2 with 39.53488372093023 % - a large (usually metallic) vessel for holding gases or liquids

1 with 36.434108527131784 % - an enclosed armored military vehicle; has a cannon and moves on caterpillar treads

3 with 24.031007751937985 % - as much as a tank will hold

- The tank is full of soldiers. (tank de luptă)

1 with 49.6124031007752 % - an enclosed armored military vehicle; has a cannon and moves on caterpillar treads

2 with 20.155038759689923 % - a large (usually metallic) vessel for holding gases or liquids

4 with 18.6046511627907 % - a freight car that transports liquids or gases in bulk

3 with 11.627906976744185 % - as much as a tank will hold

- The tank is full of nitrogen. (recipient)

2 with 53.48837209302326 % - a large (usually metallic) vessel for holding gases or liquids

1 with 35.65891472868217 % - an enclosed armored military vehicle; has a cannon and moves on caterpillar treads

3 with 6.2015503875969 % - as much as a tank will hold

4 with 4.651162790697675 % - a freight car that transports liquids or gases in bulk

- The nurse gave him a flu shot

Possible synsets

1 the act of firing a projectile

2 a solid missile discharged from a firearm

3 (sports) the act of swinging or striking at a ball with a club or racket or bat or cue or hand

4 a chance to do something

5 a person who shoots (usually with respect to their ability to shoot)

6 a consecutive series of pictures that constitutes a unit of action in a film

7 the act of putting a liquid into the body by means of a syringe

8 a small drink of liquor
 9 an aggressive remark directed at a person like a missile and intended to have a telling effect
 10 an estimate based on little or no information
 11 an informal photograph; usually made with a small hand-held camera
 12 sports equipment consisting of a heavy metal ball used in the shot put
 13 an explosive charge used in blasting
 14 a blow hard enough to cause injury
 15 an attempt to score in a game
 16 informal words for any attempt or effort
 17 the launching of a missile or spacecraft to a specified destination

Prediction:

7 with 34.883720930232556 % - the act of putting a liquid into the body by means of a syringe
 4 with 22.48062015503876 % - a chance to do something
 1 with 19.37984496124031 % - the act of firing a projectile
 12 with 12.4031007751938 % - sports equipment consisting of a heavy metal ball used in the shot put
 10 with 6.2015503875969 % - an estimate based on little or no information
 5 with 4.651162790697675 % - a person who shoots (usually with respect to their ability to shoot)

Există și varianta scurtă care întoarce doar sensul cel mai probabil. De exemplu, pentru „shot” varianta scurtă este „the act of putting a liquid into the body by means of a syringe”.

6.1.3.4. EverythingSolver

Funcția EverythingSolver generează timp de x secunde funcții matematice care explică din ce în ce mai bine datele primite. Funcția primește date de antrenare și un Int ce reprezintă numărul de secunde. CSV-ul are pe fiecare linie clasa (0 sau 1) și feature-uri.

De exemplu, am generat câteva date astfel:

```
ofstream fout("date.csv");
for(int a=0;a<20;a++)
{
    int r=rand()%100;
    if(r<50)
        fout<<"0,";
    else
        fout<<"1,";
    fout<<r<<","<<rand()%500+5000<<","<<rand()%300<<endl;
}
```

Cod 8

Generez un număr r între 0 și 99. Dacă numărul este mai mic de 50 clasa este 0, altfel este 1. Afîșez clasa, numărul r, un număr aleator între 5000 și 5499 și un număr între 0 și 299. Ultimele două numere au fost adăugate pentru a îi face mai grea munca programului. Funcția este destul de simplu de găsit (din moment ce este separabilă liniar), dar se pot găsi soluții și pentru funcții mai grele.

Un exemplu de date generate astfel:

Clasa	R	X2	X3
0	41	5334	167
0	0	5224	269
1	78	5462	258
1	64	5145	5
1	81	5461	27
1	91	5442	295
0	27	5391	36
0	4	5153	2
1	92	5421	82
0	16	5395	218
0	47	5271	126
0	38	5412	69
1	67	5035	199
1	94	5311	203
0	22	5173	33
1	64	5211	141
1	53	5047	268
0	44	5257	262
0	37	5223	259
0	41	5278	229

Cum problema este suficient de simplă am găsit o funcție care depinde doar de x_0 dar am găsit și alte soluții interesante. Când am analizat funcțiile generate, inițial nu aveau sens pentru că foloseau variabile care au fost puse să încurce. Ca notații, x_0 este primul număr de după clasă, x_1 al 2-lea etc.

Exemple de rezultate:

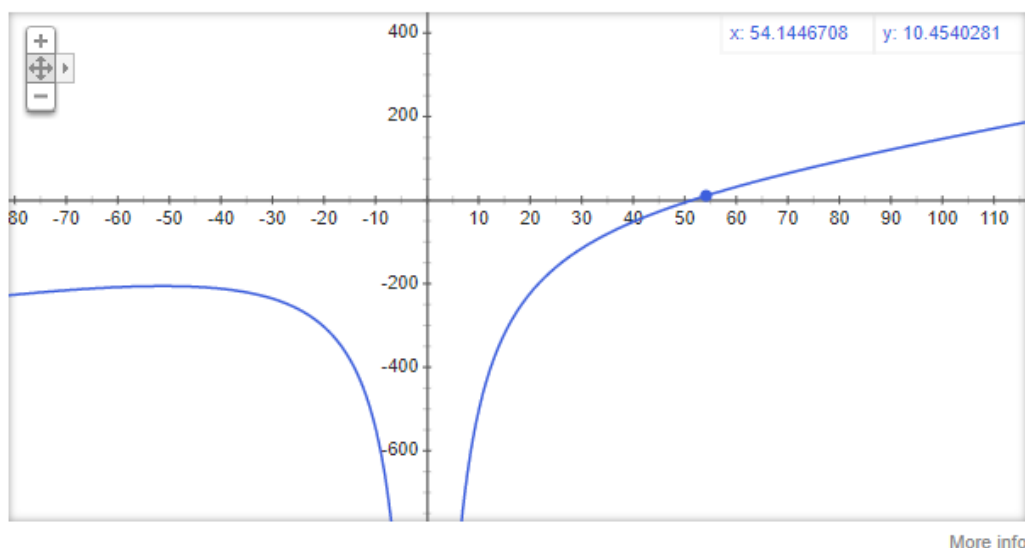
„ $2 \cdot x_0 - \text{Abs}(x_1 / (\text{Abs}(x_0) + 0.01) + 0.9) \Rightarrow 1.0$ ”

„ $100.0 \cdot x_0 - \text{Abs}(x_0 + x_1) \Rightarrow 1.0$ ”

Ca explicație, rezultatele sunt de forma „funcție \Rightarrow acuratețe”. Rezultatul funcției trebuie să fie pozitiv pentru a genera un element din clasa 1 și negativ altfel.

Dacă pentru exemplele anterioare înlocuiesc x_0 (principalul factor) cu x , înlocuiesc x_1 cu valoarea medie $5249.5 = (5000 + 5499) / 2$ și pun funcția într-un grafic se obțin[23] imaginile de mai jos, ambele trecând de la o valoare negativă la una pozitivă prin punctul (aproximativ) $x=50$.

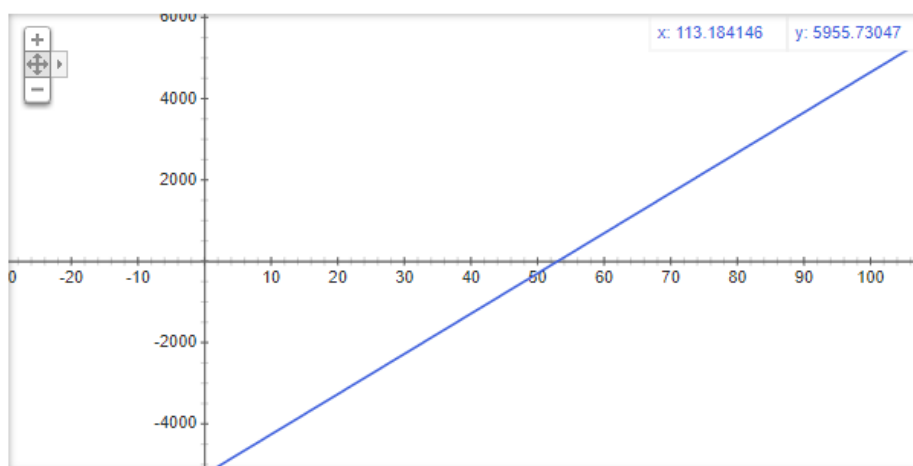
Graph for $2*x - \text{abs}(5249.5/(\text{abs}(x)+0.01))+0.9$



[More info](#)

Figura 24

Graph for $100*x - \text{abs}(x+5249.5)$



[More info](#)

Figura 25

Dacă în schimb funcția era „ $r < 25 \parallel r > 75$ then 0 else 1”, din moment ce se generează și funcții cu Abs, se poate găsi funcția „ $25 - \text{abs}(r - 50)$ ”. „25” se poate obține din „ $0.5 * 0.5 * 10 * 10$ ”.

Programul propus de mine folosește constantele [-1, 0, 0.1, 0.5, 1, 2, 10] și funcțiile [+ , - , *, / , abs]. Evident, se pot adăuga funcții mai interesante precum: sin, cos, exp, log etc., dar obținerea unui rezultat poate dura mai mult și funcțiile devin mult mai complicate.

Nu am văzut să existe sau să fie folosită o astfel de abordare (poate doar la „decision tree” este ceva asemănător) deși se poate găsi o rezolvare aproximativă (poate chiar perfectă) destul de repede.

Cod python:

```

f = open(sys.argv[1])
data = np.loadtxt(f,delimiter=",")
X_Train=data[:, 1:]
Y_Train=data[:,0]
f.close()

secun=int(sys.argv[2])
start_time = time.time()

def MyDi(a,b):
    return a/(abs(b)+0.01)
def RandomBool():
    return randint(0,1)==1
class RandomFunction:
    CateMaiSunt=0
    Variabila=True#doar daca CateMaiSunt=0
    VariabilaIndex=0
    Constanta=0
    Operatie='+'
    Left=None
    Right=None
    Feature=-1

    def MakeTerminal(self):#Tre apelat daca CateMaiSunt=0
        self.Variabila=RandomBool()
        if self.Variabila:
            self.VariabilaIndex=randint(0,self.Feature-1)
        else:
            self.Constanta=random.choice([-1,0,0.1,0.5,1,2,10])
    def MakeNeterminal(self):
        self.Operatie=random.choice(['+', '-', '*', '/', 'abs'])
        self.Left=RandomFunction(self.Feature,randint(0,self.CateMaiSunt-1))
        self.Right=RandomFunction(self.Feature,self.CateMaiSunt-1)

    def __init__(self,Featur,Cate=-1):
        self.Feature=Featur
        if Cate== -1:
            self.CateMaiSunt=randint(1,self.Feature)
        else:
            self.CateMaiSunt=Cate

        if self.CateMaiSunt!=0:
            self.MakeNeterminal()
        else:
            self.MakeTerminal()
    def __str__(self):
        if self.CateMaiSunt==0:
            if self.Variabila:
                return "(x"+str(self.VariabilaIndex)+")"
            else:
                return "("+str(self.Constanta)+")"
        if self.Operatie=='abs':

```

```

        return "(abs("+str(self.Left)+'+'+str(self.Right)+"))"
    if self.Operatie=='/':
        return "(MyDi("+str(self.Left)+'/'+str(self.Right)+"))"
    return "("+str(self.Left)+self.Operatie+str(self.Right)+")"

NrFeat=len(X_Train[0])
from sympy import *
for i in range(NrFeat):
    exec('x'+str(i)+'=symbols('\'+str(i)+'\')')
def Calc(Feat,S):
    for i in range(len(Feat)):
        S=S.replace( "x"+str(i) , str(Feat[i]) )
    return eval(S)
amaifost=set()
bestAcc=0
Total=len(X_Train)
while True:
    ElapsedTime = time.time() - start_time
    if ElapsedTime>secun or bestAcc>0.999:
        break
    RF=RandomFunction(NrFeat)
    FUNCTIE_GLOBALA=str(simplify(eval(str(RF))))
    if FUNCTIE_GLOBALA in amaifost:
        continue
    amaifost.add(FUNCTIE_GLOBALA)
    bun=0
    for i,line in enumerate(X_Train):
        R=Calc(line,FUNCTIE_GLOBALA)
        if ( R>=0 and Y_Train[i]==1) or (R<0 and Y_Train[i]==0):
            bun+=1
    NewAcc=bun/Total
    if NewAcc>bestAcc:
        print(FUNCTIE_GLOBALA,' ->',NewAcc)
        bestAcc=NewAcc

```

Cod 7

6.1.3.5. Machine learning

Am observat că în loc să fie folosit un singur algoritm de machine learning, de multe ori este mai bine să fie folosiți mai mulți algoritmi și apoi cei mai buni x să voteze. De exemplu, când am participat la un concurs am încercat principalii algoritmi și cel mai bun rezultat a fost în jur de 75%, restul fiind între 70% și 75%. Doar pentru că am ales primii 10 algoritmi și i-am lăsat să voteze, am obținut în jur de 82% acuratețe. Evident că în loc de 10 se pot alege mai mulți sau mai puțini algoritmi și voturile pot avea ponderi diferite și toate acestea se pot schimba de la o problemă la alta. Am adăugat această funcție ce permite cele menționate la sistem (funcția MachineLearning). Funcția primește un String cu algoritmi ce urmează să fie folosiți, un String care indică dacă datele să fie normalizate sau nu, un CSV cu date de antrenare și un CSV cu date de test.

Algoritmii ce pot fi folosiți în această funcție sunt:

- LR =LogisticRegression
- LDA =LinearDiscriminantAnalysis
- KNNC1 =KNeighborsClassifier(1)
- KNNC9D =KNeighborsClassifier(9, weights='distance')
- DTC =DecisionTreeClassifier(max_depth=5)
- RFC =RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1)
- MLPC =MLPClassifier(alpha=0.1)
- ABC =AdaBoostClassifier()
- GNB =GaussianNB()
- QDA =QuadraticDiscriminantAnalysis()
- GBC =GradientBoostingClassifier()
- ETC =ExtraTreeClassifier()
- BC =BaggingClassifier()
- SGDC =SGDClassifier()
- RC =RidgeClassifier()
- PAC =PassiveAggressiveClassifier()
- ETSC =ExtraTreesClassifier()
- BNB =BernoulliNB()
- GM =GaussianMixture()

Evident, se pot adăuga și alți algoritmi sau aceiași cu alți parametri dar am considerat că sunt suficienți pentru rezultate rapide și destul de bune. Fiecare algoritm este antrenat folosind 10 folduri. Pe scurt, un fold înseamnă că din datele de antrenare se exclude pe rând 10% dintre linii, se antrenează algoritmul pe restul datelor și se estimează (folosind algoritmul antrenat) clasele din care fac parte datele inițial excluse. Se obține astfel pentru fiecare algoritm o acuratețe pe datele de antrenare, iar cei mai buni 10 pot vota pentru datele de test. Pe lângă faptul că acuratețea devine mai bună folosind votul, rezultatele capătă o anumită „încredere”. Una este să fie un rezultat cu 100% din voturi și alta e să fie cu 50% (presupunând două clase).

Un exemplu clar este următorul:

Train.csv:

0	45	5238	64
1	71	5115	247
1	92	5157	11
0	5	5171	53
1	80	5030	240
1	75	5290	20
1	77	5086	198
0	4	5131	289
0	44	5110	288
0	18	5259	219

0	27	5119	209
1	79	5300	216
1	96	5458	53
0	30	5306	293
0	49	5272	154
1	60	5054	294
0	33	5229	146
1	71	5059	228
0	41	5188	21
1	66	5185	33

Test.csv:

0	41	5334	167
0	0	5224	269
1	78	5462	258
1	64	5145	5
1	81	5461	27
1	91	5442	295
0	27	5391	36
0	4	5153	2
1	92	5421	82
0	16	5395	218
0	47	5271	126
0	38	5412	69
1	67	5035	199
1	94	5311	203
0	22	5173	33
1	64	5211	141
1	53	5047	268
0	44	5257	262
0	37	5223	259
0	41	5278	229

Folosind toți algoritmi (LR, LDA, KNNC1, KNNC9D, DTC, RFC, MLPC, ABC, GNB, QDA, GBC, ETC, BC, SGDC, RC, PAC, ETSC, BNB, GM) și ambele normalizări posibile (cu și fără):

N1 LR 0.5 time: 0.07120418548583984
 N1 LDA 0.95 time: 0.05742526054382324
 N1 KNNC1 0.85 time: 0.0361940860748291
 N1 KNNC9D 0.8 time: 0.0169527530670166
 N1 DTC 1.0 time: 0.0173032283782959
 N1 RFC 0.95 time: 0.0892946720123291
 N1 MLPC 0.2 time: 0.42443227767944336
 N1 ABC 1.0 time: 0.013033628463745117
 N1 GNB 0.95 time: 0.006015300750732422
 N1 QDA 0.9 time: 0.04664349555969238
 N1 GBC 1.0 time: 0.2016303539276123
 N1 ETC 0.8 time: 0.0030105113983154297

N1 BC 1.0 time: 0.08274602890014648
 N1 SGDC 0.6 time: 0.004511594772338867
 N1 RC 0.15 time: 0.053394317626953125
 N1 PAC 0.5 time: 0.005012989044189453
 N1 ETSC 0.9 time: 0.07926607131958008
 N1 BNB 0.3 time: 0.007993936538696289
 N1 GM 0.5 time: 0.037722110748291016
 None LR 1.0 time: 0.005543231964111328
 None LDA 0.95 time: 0.007492780685424805
 None KNNC1 0.6 time: 0.010082483291625977
 None KNNC9D 0.6 time: 0.006991386413574219
 None DTC 1.0 time: 0.0030078887939453125
 None RFC 1.0 time: 0.08528709411621094
 None MLPC 0.3 time: 0.21422481536865234
 None ABC 1.0 time: 0.012028932571411133
 None GNB 1.0 time: 0.006018638610839844
 None QDA 0.85 time: 0.0049839019775390625
 None GBC 1.0 time: 0.19159960746765137
 None ETC 0.85 time: 0.00400996208190918
 None BC 1.0 time: 0.08225274085998535
 None SGDC 0.6 time: 0.004005908966064453
 None RC 1.0 time: 0.010027408599853516
 None PAC 0.5 time: 0.005014181137084961
 None ETSC 0.8 time: 0.07824254035949707
 None BNB 0.3 time: 0.00701904296875
 None GM 0.5 time: 0.01905035972595215
 0.0 100.0 %
 0.0 100.0 %
 1.0 100.0 %
 1.0 100.0 %
 1.0 100.0 %
 1.0 100.0 %
 0.0 100.0 %
 0.0 100.0 %
 1.0 100.0 %
 0.0 100.0 %
 0.0 100.0 %
 0.0 100.0 %
 1.0 100.0 %
 1.0 100.0 %
 0.0 100.0 %
 1.0 100.0 %
 0.0 70.0 %
 0.0 100.0 %
 0.0 100.0 %
 0.0 100.0 %

Datele au fost generate asemănător cu cele de la funcția EverythingSolver. Observăm că singurul rezultat greșit este acela care nu are 100% încredere (valoarea primului feature de pe acel rând este aproape de 50, limita dintre clasa 0 și clasa 1). Se mai observă și ce algoritmi sunt buni pentru o problemă anume și cât durează antrenarea/predicția față de alți algoritmi. Fiind date mici, toți algoritmi durează puțin, dar pe date mai mari se găsesc ușor algoritmi care durează foarte mult (chiar și de 200 de ori mai mult ca alții), fără să fie neapărat mai buni.

Cod python:

```
f = open(sys.argv[1])#Train
data = np.loadtxt(f,delimiter=",")
X_Train=data[:, 1:]
Y_Train=data[:,0]
f = open(sys.argv[2])#Test
data = np.loadtxt(f,delimiter=",")
X_Test=data[:, 1:]
Y_Test=data[:,0]
Functions=" "+sys.argv[3]+" "
Norm=sys.argv[4]
models = []
if "LR," in Functions:
    models.append(('LR', LogisticRegression()))
if "LDA," in Functions:
    models.append(('LDA', LinearDiscriminantAnalysis()))
if "KNNC1," in Functions:
    models.append(('KNNC1', KNeighborsClassifier(1) ))
if "KNNC9D," in Functions:
    models.append(('KNNC9D', KNeighborsClassifier(9, weights='distance') ))
if "DTC," in Functions:
    models.append(('DTC', DecisionTreeClassifier(max_depth=5) ))
if "RFC," in Functions:
    models.append(('RFC', RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1) ))
if "MLPC," in Functions:
    models.append(('MLPC', MLPClassifier(alpha=0.1) ))
if "ABC," in Functions:
    models.append(('ABC', AdaBoostClassifier() ))
if "GNB," in Functions:
    models.append(('GNB', GaussianNB() ))
if "QDA," in Functions:
    models.append(('QDA', QuadraticDiscriminantAnalysis() ))
if "GBC," in Functions:
    models.append(('GBC', GradientBoostingClassifier() ))
if "ETC," in Functions:
    models.append(('ETC', ExtraTreeClassifier() ))
if "BC," in Functions:
    models.append(('BC', BaggingClassifier() ))
if "SGDC," in Functions:
    models.append(('SGDC', SGDClassifier() ))
if "RC," in Functions:
    models.append(('RC', RidgeClassifier() ))
if "PAC," in Functions:
```

```

        models.append(('PAC', PassiveAggressiveClassifier() ))
if ",ETSC," in Functions:
    models.append(('ETSC', ExtraTreesClassifier() ))
if ",BNB," in Functions:
    models.append(('BNB', BernoulliNB() ))
if ",GM," in Functions:
    models.append(('GM', GaussianMixture() ))
Predictii=[ [] for _ in range(len(Y_Test))]
Accs=[]
normlist=[]
if Norm=="N1":
    normlist.append("N1")
if Norm=="None":
    normlist.append("None")
if Norm=="Both":
    normlist.append("N1")
    normlist.append("None")

for normalize in normlist:
    if(normalize=="None"):
        X_Test_N=X_Test
        X_Train_N=X_Train
    if(normalize=="N1"):
        X_Train_N=normalize(X_Train)
        X_Test_N=normalize(X_Test)
    for name, model in models:
        start_time = time.time()
        kf = KFold(n_splits=10,shuffle=True)
        kf.get_n_splits(X_Train_N)
        miniacc=[]
        for train_index, test_index in kf.split(X_Train_N):
            X_Tr, X_Te= X_Train_N[train_index], X_Train_N[test_index]
            Y_Tr, Y_Te= Y_Train[train_index], Y_Train[test_index]
            model.fit(X_Tr,Y_Tr)
            miniacc.append( (model.predict(X_Te)==Y_Te).mean() )

        model.fit(X_Train_N,Y_Train)
        Preds=model.predict(X_Test_N)
        acc=(np.array(miniacc)).mean()
        print(normalize,',',name,',',acc,' time: ',time.time() - start_time)

    if len(Accs)<=10 or acc>=np.array(Accs).mean():
        Accs.append(acc)
        for i in range(0,len(Preds)):
            Predictii[i].append(Preds[i])

BestIndex=np.array([x for x in Accs]).argsort()[::-1][1:10]
for i in range(len(Predictii)):
    Pred=np.array(Predictii[i])[BestIndex]
    print(Counter(Pred).most_common(1)[0][0],',',100.0*Counter(Pred).most_common(1)[0][1]/(len(Pred)),'%')

```

Cod 8

Amintesc că pentru a folosi acești algoritmi de IA avem nevoie de cât mai multe date. Acestea se pot obține și de pe internet cu funcția JsOnUrl și se pot folosi instanțe automate.

6.2 Exemple și funcții pe care nu le-am implementat încă

6.2.1. Basic

- Generarea unui număr aleator între 0 și 1;
- Trimiterea mail-urilor;
- Map, Fold.

6.2.2. File transform

O funcționalitate de care am avut nevoie a fost „Files downloader”. Un program care descarcă de pe un link toate fișierele de un anumit tip. De exemplu, PDF-urile de pe pagina <http://old.unibuc.ro/~lleustean/Teaching/2016-TCO> .

Am vrut să adaug mai multe transformări între tipuri similare de date. De multe ori am avut nevoie să transform o arhivă din .rar în .zip (deoarece nu era instalat Winrar sau foloseam Linux). Pentru clasificarea imaginilor sau alte probleme legate de vision, pe lângă aducerea la aceeași rezoluție, trebuie aduse toate fișierele la același tip de date (.bmp, .jpg, .png etc.) pentru a fi prelucrate mai ușor. Pentru arhive am găsit ArcConvert[24] care are foarte multe opțiuni.

Am observat că sunt multe fișiere de tip audio și video care sunt stereo și folosesc doar unul dintre cele două canale disponibile. Acest lucru se poate rezolva folosind programul ffmpeg („ffmpeg.exe -i .\fisier.mp3 -ac 1 mono.mp3” respectiv „ffmpeg.exe -i .\fisier.mp4 -ac 1 mono.mp4”). În loc să căutăm un program care rezolvă această problemă, apoi să observăm că programul este pentru Linux în loc de Windows și în cele din urmă să descărcăm un program de editare gratuit (precum Audacity[25]) care o să fie folosit o singură dată, se poate căuta funcția care primește un Mp3 și întoarce un Mp3 (probabil nu sunt multe).

Adăugând posibilitatea de a pune texte și imagini (la anumite coordonate) pe imagini, se poate automatiza completarea mai multor formulare. De exemplu, pentru a pune o imagine (poză sau imagine generată pe calculator) pe site-uri de stock, este nevoie să fie completat un formular cu data curentă (text) și imaginea respectivă la anumite coordonate. Asta poate să fie foarte ușor automatizat cu cele două funcții menționate.

6.2.3. Inteligența artificială

După ce sunt extrase feature-uri din imagini, text etc. pentru a folosi algoritmi de ML este posibil să apară duplicate. De multe ori am avut nevoie să folosesc un program care să elimine liniile care se repetă.

Când am făcut un program care detecta obiecte folosind camera web, a trebuit să descarc de pe Google multe imagini cu acel obiect. Am încercat extensii ale browser-ului care descarcă automat toate imaginile de pe o anumită pagină. Ori descărca în plus, ori nu deschidea imaginile

înainte să le salveze și aveau rezoluție mult mai mică. O funcție specializată care primește un String și întoarce o listă de imagini cred că e ușor de făcut și destul de utilă.

O altă funcție interesantă constă în extragerea unei părți din înregistrări audio sau video. De exemplu, când apare un obiect în cadru (în cazul filmelor) sau când este spus un anumit cuvânt. O astfel de funcție s-ar putea găsi ușor deoarece nu sunt multe care primesc un video, un String, un Int și întorc o listă de video-uri (Int-ul semnifică ce dimensiune are fereastra, câte secunde înainte și câte după ce este rostit cuvântul respectiv). Această funcție poate să fie folosită pentru a genera date de antrenare, extrăgând din filme sau înregistrări lungi secvențe de sunete utile. De exemplu, funcția poate să fie folosită pentru a face un program care clasifică anumite cuvinte diferite dar care sună asemănător (to/too/two, loose/lose, affect/effect) sau pentru a testa cât de bine se descurcă pe aceste date un astfel de program. Sunt și persoane care fac compilații cu aceste secvențe pentru că sunt interesante ca atare[26; 27], ceea ce poate dura destul timp chiar și în condițiile în care sunt deja notate momentele în care apare cuvântul respectiv.

De multe ori datele trebuie normalizate. Doar transformând datele și aducându-le în intervalul [0,1] se pot obține rezultate mult mai bune. Dacă nu sunt normalizate datele, un KNN ar putea crede că un feature este mult mai important față de celelalte doar pentru că are valori mai mari. Cea mai simplă normalizare folosită de mine este „ $x=(x-\min)/(\max-\min)$ ”, unde min și max sunt valorile minime și maxime ale acelui feature. Evident, nu are sens dacă toate valorile sunt egale. Aceleași numere trebuie păstrate (min și max) și pentru următoarele date de intrare (chiar dacă valorile pot ieși din intervalul [0,1]). Deci ar fi utilă o funcție care primește trei Double-uri (min, max și x) și întoarce valoarea normalizată.

O altă funcție utilă pentru vision este cea care extrage imagini din video-uri (la fiecare x secunde sau x frame-uri echidistante). De exemplu, funcția este utilă unui clasificator ce distinge între două obiecte. În loc să fie făcute zeci de poze din diferite unghiuri, este mult mai ușor să înregistrezi video obiectul respectiv și apoi să folosești funcția care extrage imagini. Acele imagini pot fi utile în sine (pentru a face un alt video de tip „stop motion” sau pentru a avea un „preview” al acelui video) dar și pentru alți algoritmi de AI (din poze se pot extrage feature-uri ce pot clasifica videouri în filme color/alb-negru, animații/videouri filmate, acțiune/comedie etc.).

Alte programe de OCR arată și poziția textului, sau chiar pozițiile obiectelor din imagine [28].

WSD se folosește de eliminarea cuvintelor de legătură (stopwords), funcții care întorc sensurile posibile ale unui cuvânt și funcții care returnează definiția unui sens. Aceste funcții se pot adăuga astfel încât WSD să fie făcut din funcții Somon.

Utilă poate să fie și o funcție care face traduceri între diferite limbi. Se poate folosi pentru a extrage un text dintr-o imagine și apoi tradus textul în altă limbă pentru a face text mining pe imagini. Aceste imagini pot avea textul în altă limbă față de limba pentru care a fost scris programul de text mining (de exemplu engleză).

După ce s-a obținut un text, acesta poate să fie destul de lung. Aș adăuga și posibilitatea de a face un rezumat al unui text. Am găsit două aplicații care par interesante și una se poate deja implementa din moment ce este scrisă în Python [29; 30].

Ultima idee notată a fost aceea de a detecta câte persoane vorbesc într-o înregistrare. Apoi, pentru fiecare persoană, se pot aplica algoritmi de author profiling și se pot afla: tipul de personalitate, nivelul de educație al persoanei, vârsta, genul etc. [31].

7. Provocări

Prima provocare a fost să fac totul cât mai intuitiv, de la modul de funcționare și până la modul în care sunt afișate nodurile atunci când este vizualizată o funcție.

Am încercat ca modul de folosire să fie la fel atât pe telefon cât și pe calculator (de exemplu nu am folosit click dreapta și alte limitări de acest tip).

Unele funcționalități au fost mai greu de implementat față de estimările mele, de exemplu am reușit să adaug funcționalitatea Filter după 3 încercări.

8. Programe similare

Nu am găsit un program care să aibă toate funcționalitățile pe care le propun, sunt puține asemănătoare și foarte puțin populare, dar au unele idei bune ce pot fi adăugate ulterior la acest proiect.

Când am căutat „dataflow” sau „pipeline”, primele rezultate erau pentru programe de UML pentru reprezentări sau erau de la un Cloud (deci cu bani) și erau foarte specifice (doar pentru SQL-uri).

O aplicație similară cu ce mi-am propus să fac în această lucrare a fost „MIT App Inventor” [14], aplicație găsită din întâmplare, dar aceasta este doar pentru Android.

Un alt program simiar este Flowgorithm [15] și se pot face unele comparații între Somon și acest program. Despre Flowgorithm:

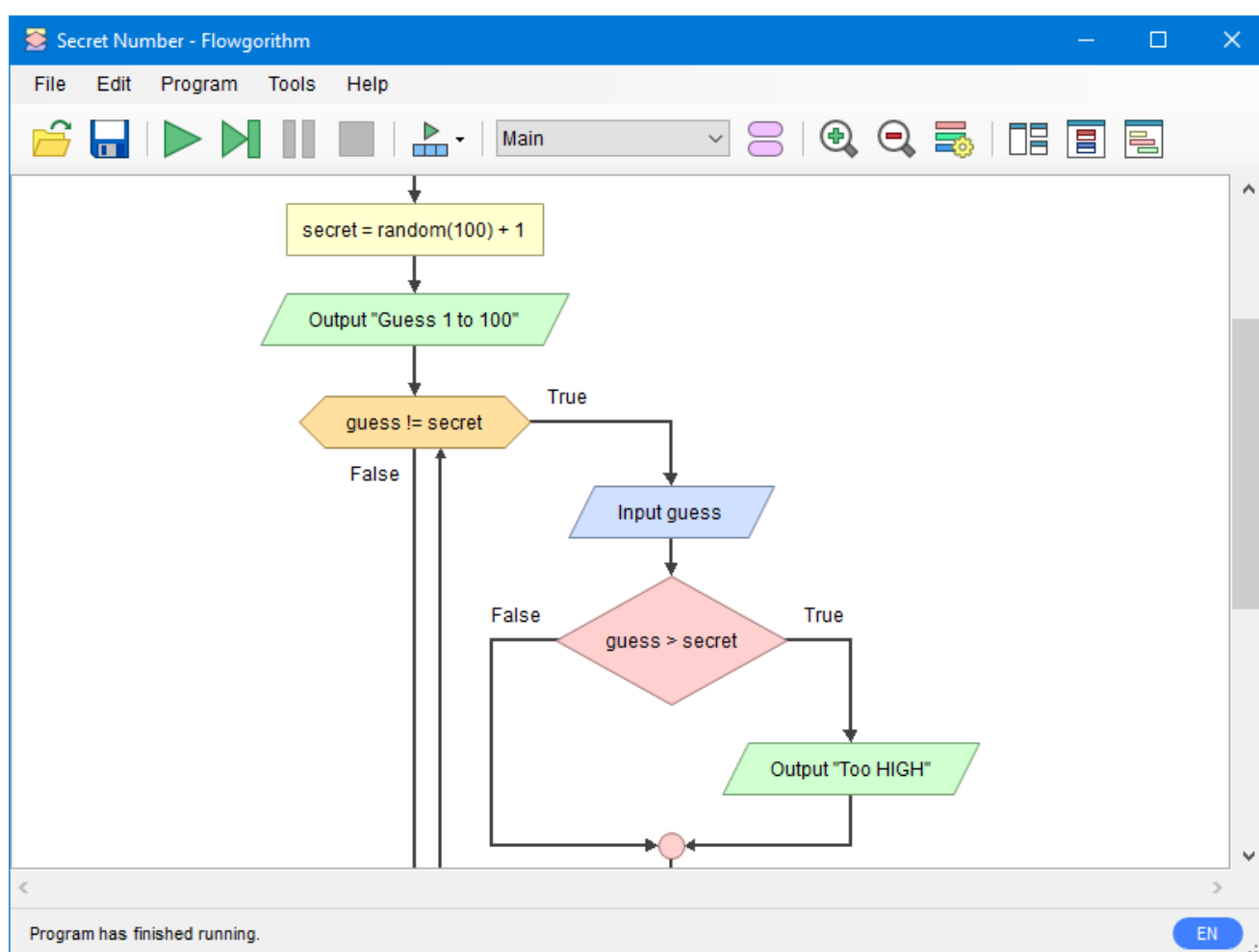


Figura 26

- Codul sursă nu este public, din cauza asta nu pot fi reparate buguri, nu pot fi adăugate funcționalități noi etc. De aceea ești obligat să folosești cele 22 de funcții default puse la

dispoziție (dintre care 12 sunt de matematică, 2 legate de stringuri, 6 legate de conversii, random și dimensiunea unui array);

- Nodurile au o culoare și o formă anume pentru fiecare tip de nod;

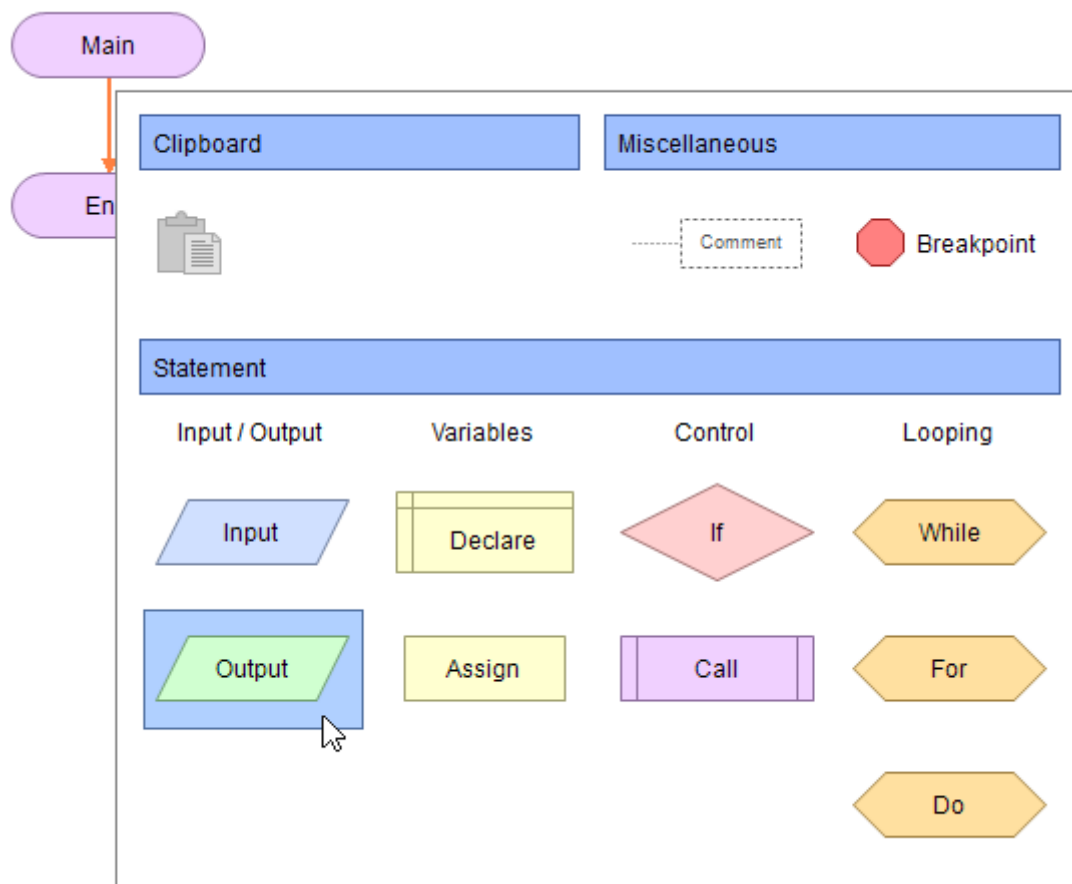


Figura 27

- Sunt doar 12 exemple de programe;
- Este doar pentru Windows;
- Are translator (generator de cod în 21 limbaje de programare) dar asta limitează adăugarea unor funcționalități noi precum operațiuni legate de fișiere;
- Permite rularea programului direct din aplicația de Windows.

9. Direcții de îmbunătățire

În forma curentă, nu este un program foarte eficient, nici foarte sigur și au mai rămas multe funcționalități de adăugat.

9.1. Utilizarea SC

În funcție de cât și cum e folosit acest sistem, se pot schimba ratele de câștigare/consumare a SC. De exemplu, acum se acumulează 1 SC pentru fiecare minut de minat și se consumă 1 SC pentru orice instanță pe care o minează altcineva (indiferent cât durează). În principiu, se pot face și licitații dacă ești dispus să dai un număr mai mare de SC-uri pentru a fi rezolvată cât mai repede o instanță. Sau să se schimbe numărul de SC-uri consumate, ținând cont de funcțiile default folosite (să accesezi un site este mai periculos pentru un miner față de a calcula suma a două numere).

Se mai pot trimite SC-uri către cei care au făcut funcțiile, atunci când acestea sunt apelate pentru a încuraja utilizatorii să facă funcții cât mai diverse, utile și simple.

9.2. Ordinea executării

O funcționalitate importantă care există la limbajele clasice procedurale este siguranța ordinii executării instrucțiunilor. Ceva similar se poate implementa și aici. De exemplu, se pot pune pe noduri ordinea/prioritatea executării funcțiilor asociate acestora. De exemplu, dacă un program trebuie să trimită și un SMS și un Mail, să se poată alege ca întâi să fie trimis Mail-ul și apoi SMS-ul.

9.3. Translator pentru alte limbaje

O funcționalitate interesantă și aparent utilă este transformarea unui cod scris într-un limbaj de programare (de exemplu C++) într-o funcție Somon (sau invers). Acest lucru ar limita funcționalitatea Somon-ului deoarece nu toate limbajele au aceleași funcționalități (de exemplu Goto), iar multe limbaje sunt încă în schimbare, având funcționalități noi în fiecare an. Cred că unele funcții simple pot fi „traduse” în funcții Somon și invers dar, pentru aplicații mari ce folosesc variabile globale, goto-uri și librării mari, pare destul de greu de realizat.

9.4. Variantă locală

Un dezavantaj al programului este acela că nu poate interacționa cu periferice sau cu utilizatorul. Deci nu se poate folosi camera web pentru a înregistra/detecta anumite evenimente, imprimanta pentru a printa un eventual rezultat, cititoare de carduri etc.

Deocamdată, nu se poate folosi programul pe un calculator fără acces la internet sau dacă nu vrei să uploadezi fișiere mari ca apoi să le descarci tot tu, să le prelucrezi și apoi să uploadezi rezultatul ca să îl descarci iar.

În concluzie, o variantă locală a programului este utilă în aceste cazuri.

9.5. Optimizări

Se pierde mult timp cu funcționalități care nu sunt esențiale pentru sistem. De exemplu, numărarea de apelări de funcții pentru statistici, salvarea stării finale (numărul de obiecte pe porturi la sfârșitul instanței), upload și download fișiere intermediare etc.

Ținând cont de funcția publică, este eficient să aleagă cel care a făcut funcția respectivă ordinea executării nodurilor (în cazul în care sunt mai multe noduri de ieșire). Poate fi mai bună o abordare BF, DF sau chiar una personalizată pentru funcția respectivă.

9.6. Securitate

Acesta este unul dintre principalele dezavantaje ale sistemului, ținând cont că poți descărca fișiere necunoscute și poți uploada fișiere personale importante. Se pot pune avertismente sau setări pentru a opri anumite instanțe atunci când sunt folosite funcții default care accesează site-uri sau folosesc fișiere.

Nu ajută (sau este mult mai complicat și ineficient) să fie criptate fișierele deoarece de cele mai multe ori datele trebuie să ajungă cândva „în clar” (de exemplu pentru OCR).

9.7. Paralelizabil

Este evident că sistemul este paralelizabil, dar nu este mereu clar când trebuie să intervină multithreading-ul. Crearea și distrugerea thread-urilor pentru funcții foarte mici (mutarea unei valori de la o variabilă la alta sau suma a două numere) poate dura uneori mai mult decât rezolvarea funcției în sine. Și asta doar pe un calculator. Nu este mereu clar când este mai bine ca o funcție să fie rulată pe un alt calculator cu resurse mai bune și apoi întors rezultatul pentru rezolvarea instanței (deoarece asta depinde de mulți factori și aspecte precum viteza internetului la acel moment, eventual costul folosirii internetului dacă abonamentul se măsoară astfel etc.) și implică noi probleme (datele se trimit cu erori, dispare conexiunea la internet pentru unul dintre cei 2 mineri etc.). Cu toate acestea, se pot calibra factorii descriși pentru a produce un sistem foarte eficient (folosind datele de la cât mai multe instanțe).

9.8. Setări

Multe aplicații ce folosesc putere de procesare pentru diverse proiecte științifice oferă posibilitatea de a alege ce procente din RAM, placă video sau procesor sunt folosite. Ceva similar ar ajuta acest proiect.

9.9. High order functions

Multe funcții se pot simplifica folosind una (sau mai multe) dintre funcțiile Map, Filter și Reduce. Acestea sunt funcții de nivel înalt (high order functions). Ele se găsesc în multe limbaje de programare (eventual cu nume schimbat, de exemplu Fold în loc de Reduce sau Transform în loc de Map) precum Python, C++, Javascript, Haskell etc.

Pe scurt, Filter primește o listă cu obiecte de un anumit tip și o funcție ce primește un element de acel tip și întoarce un Bool. Filter filtrează elementele din listă și le păstrează doar pe cele care îndeplinesc proprietatea respectivă. De exemplu, se pot filtra numerele de la 1 la 100 cu funcția Prim. Deci rezultatul este de același tip, eventual cu mai puține elemente. Antetul în Haskell este: `filter :: (a -> Bool) -> [a] -> [a]`.

Map primește o listă și o funcție iar rezultatul este obținut prin aplicarea funcției asupra fiecărui element din listă. Antetul este `map :: (a -> b) -> [a] -> [b]`. De exemplu, se poate transforma o listă de fișiere Mp3 în listă de fișiere Wav.

Fold reduce o listă la o singură valoare. Primește o listă de tip a, o funcție și o valoare de start (de tip b). Antetul este `fold :: (a -> b -> b) -> b -> [a] -> b`. Cel mai simplu exemplu este suma elementelor unei liste (unde funcția este „+” iar valoarea de start este 0) sau produsul lor (unde funcția este „*” iar valoarea de start este 1).

Aceste funcții mi se par foarte utile pentru că sunt simple de folosit și apar destul de des probleme care se rezolvă cu Map, Filter și Fold. Am implementat până acum doar Filter pentru a arăta că Somon-ul permite astfel de funcții, dar Map și Reduce sunt la fel de utile.

9.10. Mai multe statistici

Se pot adăuga mult mai multe statistici pentru fiecare utilizator și statistici globale (pentru tot sistemul) și se pot genera grafice utile de pe urma lor:

- Numărul de instanțe rezolvate pe zi/oră, numărul de funcții create etc.;
- Cele mai folosite funcții default (în funcții publice create, numărul de apelări în rezolvarea instanțelor);
- Cât de utilizate sunt 2 funcții alăturate în funcțiile publice, eventual împărțite pe categorii (def->def, pub->pub, def->pub, pub->def).

9.11. Simplificarea utilizării

Se pot face unele setări pentru a face utilizarea cât mai simplă. De exemplu, ascunderea claselor, recursivității, instanțelor automate, filter etc.

Rezultatul funcționalității „Guess function” este momentan de tip text, se poate relativ simplu genera funcția Somon respectivă.

Se pot adăuga comentarii pe nodurile funcțiilor.

9.12. Erori

Erorile pot fi mai bine semnalate, față de ce se întâmplă acum. Nici împărțirea la 0 nu primește măcar un warning, aplicația da crash și instanța rămâne neterminată. Singurul caz tratat este cel în care nu se mai mișcă date pe porturi și instanța dă în acest caz fail. Eventual se poate afișa și motivul pentru care instanța s-a oprit. Se pot introduce eventuale noduri de erori pentru a opri o instanță dacă parametrii de intrare nu îndeplinesc anumite proprietăți. Dacă o subfuncție eșuează, să fie transmisă o eroare cât mai explicativă pentru a urmări problema până la funcția

instanței. Se poate păstra o eventuală conexiune cu serverul aplicației pentru a verifica dacă minerul încă mai merge, are conexiune la internet etc., pentru a nu îl ține în așteptare pe utilizatorul care a adăugat o instanță ce nu mai poate fi rezolvată.

9.13. Tipuri noi de date

Se pot adăuga alte tipuri de date (pentru diferite tipuri de fișier de exemplu) dar sunt și alte moduri de a folosi tipurile existente. De exemplu, pentru constante se poate adăuga posibilitatea de a alege de câte ori să fie folosite valorile de acolo în loc să fie generate la infinit. Au fost multe momente când era mai simplu să folosesc o constantă o singură dată.

9.14. Altele

Funcțiile publice sau default ar putea avea valori implicite pe anumite porturi. Un auto-updater pentru miner este foarte util (când apare o funcție default nouă sau când se repară bug-uri). Se poate integra cu alte limbaje (de exemplu o librărie pentru C++) pentru apelări de instanțe și obținerea rezultatului instanței. În plus, se pot adăuga animații care arată modul în care a fost rezolvată instanța.

Guess function from example

O altă funcționalitate poate să fie folosirea exemplelor pentru a găsi noi funcții. De exemplu, pornind de la „[4,7,19,2011] => [False, True, True, True]” sistemul poate să încerce diferite funcții și să găsească o potrivire cu un Map peste funcția publică Prime. Nu este o idee bună să fie introduse și constante (pentru un eventual „Map > 5”) deoarece ar introduce prea multe calcule/posibilități. Cel puțin o încercare de funcții default/publice (fără combinații de funcții) este destul de ușor de implementat.

10. Concluzii

În perioada realizării acestui proiect, am găsit numeroase probleme care se puteau rezolva foarte ușor cu Somon-ul, dacă erau implementate funcțiile respective.

Nu am pornit cu toate acestea în minte când am început proiectul, dar mă bucur ca le-am găsit (și în opinia mea rezolvat):

Example

Am văzut zeci de proiecte bune care nu erau bine explicate și nu aveau exemple (lucru ușor de adăugat în această aplicație). Am observat că cele mai utile răspunsuri la problemele de programare erau cele care aveau cel puțin un exemplu (pentru web, există jsfiddle[3] care memorează 2-3 fișiere și permit simularea soluțiilor direct online).

Puzzle

Mi-am dat seama că adăugând piese mici (funcții) pot ajuta la simplificarea și creerea noilor funcții care ar fi foarte complexe (exemplul meu preferat este Mp3 => partituri).

Dependințe

Am văzut că multe proiecte bune ajung să nu fie folosite pentru că nu sunt ușor de utilizat sau au multe dependințe (un anumit limbaj sau anumite librării din acel limbaj). Puțini o să instaleze Python și toate dependințele necesare pentru a avea un program ce primește un film și întoarce subtitrările generate automat, deși sunt mulți oameni care au nevoie de această facilitare[8].

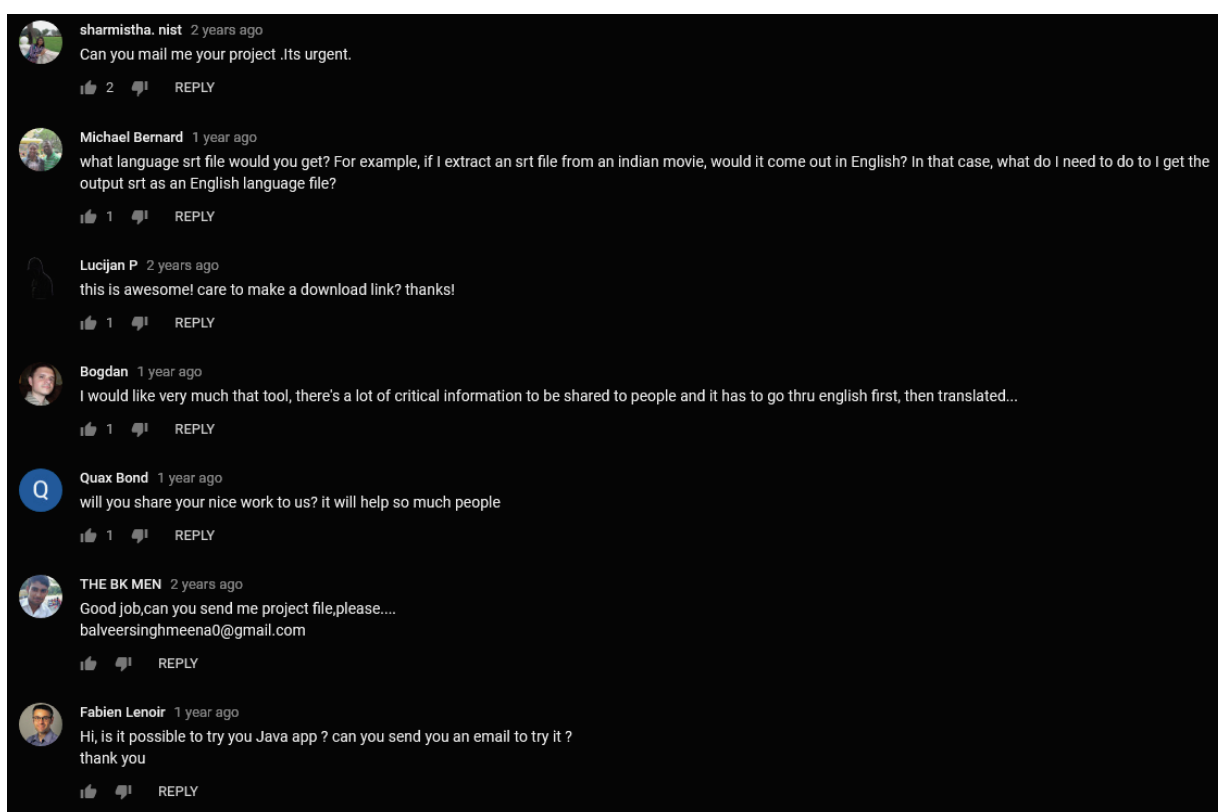


Figura 28

Paralelizare

Am văzut multe proiecte (și personale și pentru școală) care ar beneficia de această împărțire a taskului pe mai multe calculatoare pentru a scurta timpul de execuție. Tot aici intră și folosirea eficientă a calculatorului. Dacă sunt suficiente calculatoare deschise în lume în orice moment, nu trebuie să ai neapărat deschis calculatorul 24/7 ca să salvezi o valoare de pe un site o dată la 30 minute.

Portabilitate

Am observat cât de bine e să nu depinzi de un tip de dispozitiv/sistem de operare/IDE când programezi și să fie la fel de ușor de programat de pe telefon, tabletă, calculator etc.

Utilizare simplă

Se pot face funcții foarte ușor fără a fi necesar să știi să programezi (sau să folosești anumite software-uri specifice, mai mult sau mai puțin gratuite). De exemplu, pentru „Mp3 => partituri” poți să pornești cu o variabilă Mp3 și din aproape în aproape să construiești funcția dorită (funcțiile disponibile sunt filtrate ținând cont de datele de intrare și de ieșire ale nodului respectiv). Sau direct poate fi folosită funcționalitatea Guess funcion. Funcțiile au definite clar valorile de intrare și cele de ieșire.

Flow

Am folosit 3 programe diferite pentru exemplul „Mp3 => partituri” (ffmpeg[5], waon[6], sheet[4]). Dacă aveam un număr mare de fișiere audio pentru care trebuia să fac manual această transformare, aș fi scris un program care să folosească aceste trei software-uri.

Refolosire

Acel program menționat mai sus, probabil ar rămâne nefolosit după terminarea task-ului, eventual șters sau pierdut și rescris peste un anumit timp. Alte persoane în aceeași situație ar face același lucru și cei care nu știu să programeze sunt forțați să facă această muncă manual. Făcând funcția respectivă publică, dispare tot acest timp în care se verifică dacă funcția există și nu este nevoie de scrierea unui program în zeci de limbaje, în zeci de moduri. Este important faptul că se adaugă o funcție ce poate să fie folosită într-un context mai larg.

Limitări

Sunt suficiente moduri de a transforma diferite date (atât pe internet cât și software-uri trial/free, iar acele software-uri pot intra ușor în partea de **Dependințe** menționată anterior), dar de multe ori au anumite limitări. Poți să folosești programul de un anumit număr de ori, pot să îți apară watermark-uri sau ceva asemănător în rezultat, ai un PDF de 21 Mb iar limita este de 20 Mb etc. Aceste limitări au sens când ai multe fișiere de procesat, dar când ești dispus să îți procesezi propriile date, aceste limitări nu trebuie să existe.

Securitate

Acele software-uri despre care vorbeam nu oferă siguranță dacă nu sunt open-source. Orice fișier poate foarte ușor să fie trimis oriunde (iar deconectarea de la internet de fiecare dată când folosești un program nu e chiar cea mai bună soluție). De aceea, orice funcție publică are

atașată lista cu dependențe astfel încât atunci când folosești o funcție poți vedea ce este folosit în spatele ei (funcția Even nu trebuie să folosească ExecuteJsOnUrl).

Snippets (Snippet is a programming term for a small region of re-usable source code, machine code, or text)[7]

Pe lângă exemple ale unor instanțe, se pot face și funcții ce pot fi trimise altor utilizatori. Astfel, se poate trimite ușor o funcție simplă ce poate să fie testată și analizată fără să fie nevoie să folosească ambele persoane același limbaj de programare.

Funcțiile bune sunt răsplătite

Dacă o funcție este foarte des utilizată, clară, simplă sau eficientă, aceasta iese în evidență deoarece funcțiile pot fi votate (cu SC). Aceasta ajută și la partea de **Securitate** menționată anterior (o funcție cu foarte multă susținere este mai puțin probabil să fie rău intenționată). Astfel rămân doar funcții cu un nume clar, o descriere bună și cu multe exemple.

„Cod” mai clar

Se pot identifica mai ușor, vizual, bucăți de cod ce pot fi inutile pentru funcția respectivă. De exemplu, pentru următoarea funcție se declară un vector, se parcurg 1000 de numere, se fac 1000 de comparații, fără să afecteze rezultatul funcției. Exemplul de mai jos este puțin exagerat dar am întâlnit funcții de sute de rânduri care conțin cod complet inutil (cod care a rămas în urma unor editări) și devine mult mai greu de urmărit dacă acel cod este util, cu cât numărul de rânduri este mai mare. În imagine se observă destul de clar că sunt două zone complet distincte.

```
Int f(int n)
{
    vector<int> inutil;
    for(i=1; i<1000; i++)
        if(i>500)
            inutil.push_back(i);
    return n%2==0;
}
```

Cod 9

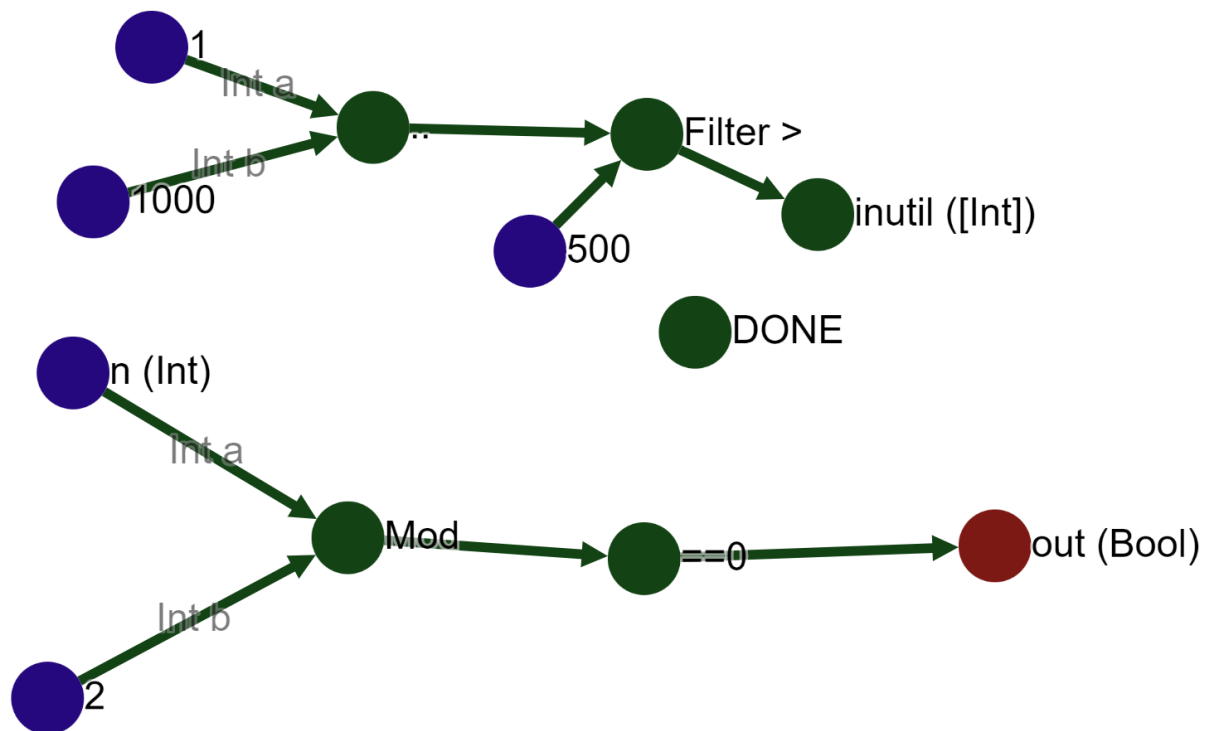


Figura 29

Alt pattern este cel în care sunt atribuiri în plus, care amestecă linii ce pot fi separate sau puse în altă ordine pentru a face totul mai clar. De exemplu, după atribuirea lui B, deci pe ramura de sus urmează în ordinea citirii, atribuirea lui E, de pe ramura de jos, și apoi o altă atribuire pe partea de sus. Evident este exagerat și acest exemplu, acest pattern devine mult mai greu de depistat în cazul funcțiilor mai mari, doar uitându-te la text. Iar adăugarea liniilor de cod într-o funcție se întâmplă destul de des.

```

Int f()
{
    A=3
    D=2
    B=A
    E=D
    F=A
    G=D
    R=B+A
    H=G
    Q=R+H
    return Q;
}
  
```

Cod 10

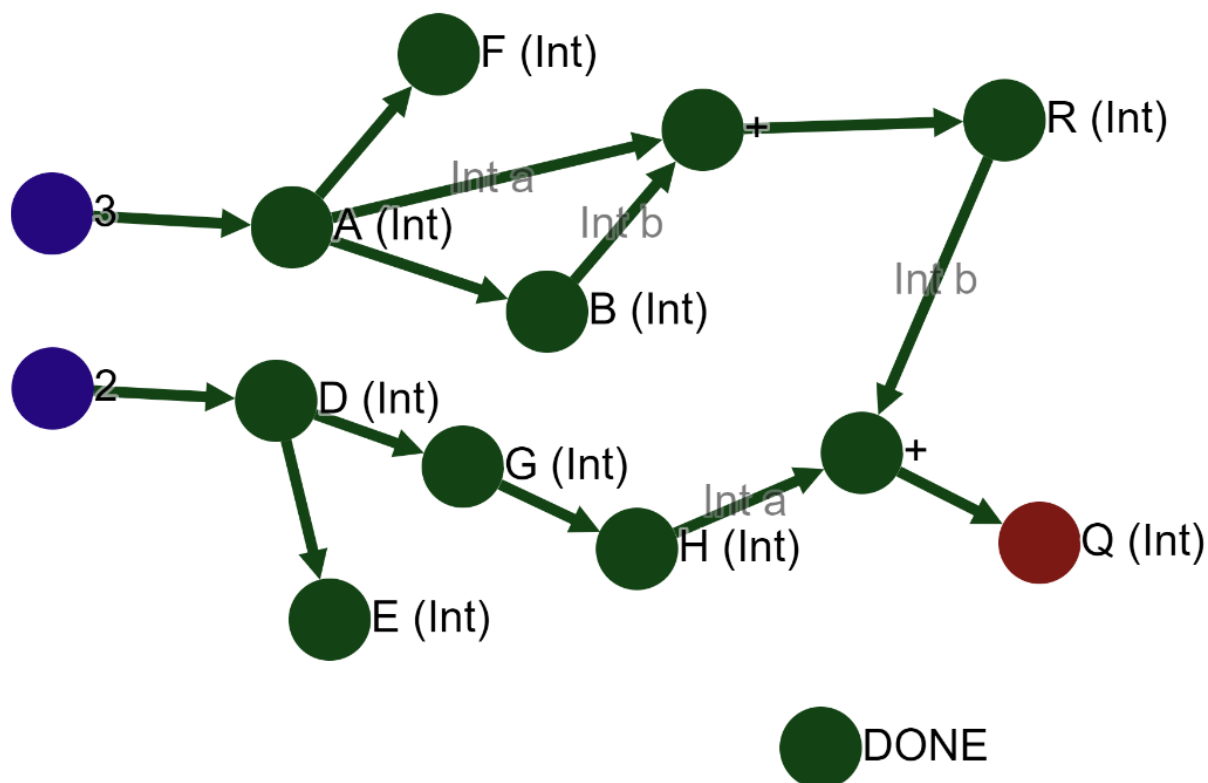


Figura 30

Se observă mult mai ușor că E și F sunt inutile iar G și H, chiar dacă sunt folosite, ele doar complică funcția fără a fi utile.

Am avut nevoie de multe ori să mut o linie mai sus sau mai jos (pentru claritate de exemplu) într-o funcție scrisă în C++ și nu era mereu clar/simplu de văzut dacă afectează cu ceva restul codului.

Tot legat de exemplul de mai sus, este mult mai ușor de văzut dacă o funcție poate avea părți rulate în paralel. Se vede clar că până la nodurile R și H totul este independent. La fel de evident este și pentru alte funcțiile mai mari/complex.

Stocarea puterii computaționale

Un avantaj al folosirii Somonului este acela că se pot acumula SC-uri în timp și că poate fi folosit atunci când este nevoie. Spre exemplu, firmele care au oferte de Black Friday și au nevoie timp de câteva zile pe an de mai multă putere computațională. Pentru cercetare, când poți să donezi timp atunci când nu sunt proiecte active, iar când ai ceva urgent de făcut, pe lângă calculatoarele disponibile poți cere ajutor de la acest sistem.

Este pentru oricine

Se poate folosi și low level, când vrei să îți faci propria funcție mai rapidă, dar la fel de bine poate fi folosit sistemul și la nivel înalt, pentru a rula instanțe folosind funcții deja făcute (sau eventual să se creeze funcții simple de 2-3 noduri pentru cei mai puțin interesați de programare).

Flexibil/adaptabil

Din moment ce codul este open-source, oricine poate să modifice funcțiile de bază, interfața, modul de rulare al miner-ului (eventual să-l facă mai eficient), obținându-se astfel un limbaj diferit.

Avantajele funcțiilor structurate

Deoarece în sistem funcțiile sunt reprezentate prin grafuri și sunt memorate multe funcții, de acolo se pot extrage informații utile. De exemplu, dacă nu ar fi definită funcția Even și ea ar fi folosită foarte des, am putea să căutăm ce noduri și ce muchii sunt des folosite împreună și am putea găsi că după „Mod” urmează de multe ori „=0” (iar „Mod” primește constanta 2). Alte informații pot fi măsurători despre cât de complicată este o funcție (cum există pentru limbajele standard LOC - lines of code - sau numărul de variabile etc.), numărul de noduri, numărul de muchii, numărul de muchii împărțit la numărul de noduri, distanța celui mai lung lanț, tipul variabilelor etc. Această simplitate poate fi un mod de a căuta/favoriza o funcție față de celelalte. Schimbând modul de programare apar și funcționalități noi precum „Guess”, „Simplify”, posibilitatea de a filtra funcțiile disponibile unui nod în funcție de context etc.

Aceste probleme sunt destul de des întâlnite

Căutând CLI-uri (Command Line Interface, fișiere executabile ce nu folosesc nici o interfață) utile pentru Somon, am dat peste acest comentariu care rezonază cu acest proiect [11]:

„So many different command line processing libraries out there and none of them just work! Some bring their whole extended family of related and unrelated external dependencies (yes, I'm looking at you Boost). Some require quirky syntax and/or very verbose setups that sacrifice simplicity for the generation of a cute usage message and validation. Many come to dominate your main() file and yet others do not build on multiple platforms - for some even their own tests and trivial usage cause crashes on some systems. Argh!”

Dezavantaje:

- Încă nu sunt implementate anumite funcții (dar sunt programatori care implementează algoritmi în cât mai multe limbaje[9]);
- Utilizatorul depinde de un server permanent funcțional (nu și dacă utilizatorul ține propriul server sau folosești „Local Somon”);
- Este greu de făcut debug când greșești o funcție sau dai de o excepție (cel puțin până când a apărut funcționalitatea „End state”);
- În afară de această lucrare de disertație, nu există o documentație clară a programului Somon și nu a fost testat pe scară largă;
- Ideea de Dataflow exista încă din anii '60; probabil sunt și alte motive pentru care acest mod de a programa nu a fost mai des utilizat până acum.

Limbaj înalt, de viitor

Limbajele de programare au tot evoluat, mereu spre limbaje mai înalte. De la cod mașină, la limbaj de asamblare, la C, apoi la C++. De atunci lucrurile au rămas cam la fel. Fără să îmi

propun neapărat, am atins cu acest proiect câteva dintre tendințele curente în evoluția limbajelor de programare [10]:

„-Increasing support for functional programming in mainstream languages used commercially, including pure functional programming for making code easier to reason about and easier to parallelise (at both micro- and macro- levels)

-Constructs to support concurrent and distributed programming.

-Mechanisms for adding security and reliability verification to the language: extended static checking, dependent typing, information flow control, static thread safety.

-Alternative mechanisms for composability and modularity: mixins, traits, typeclasses, delegates, aspects.

-Component-oriented software development.

-Increased interest in distribution and mobility.

-Open source as a developmental philosophy for languages, including the GNU Compiler Collection and languages such as Python, Ruby, and Scala.

-Massively parallel languages for coding 2000 processor GPU graphics processing units and supercomputer arrays including OpenCL

-More interest in visual programming languages like Scratch”

Pe scurt:

Folosind în programare Meta-date despre funcții, pot să apară funcționalități noi cum ar fi generarea de funcții pornind de la tipul datelor de intrare și alegând tipul de date al rezultatului (Guess function) sau aceea de a nu mai căuta o funcție anume prin zeci de fișiere (putem filtra în funcție de datele de intrare/ieșire). Principalele avantaje ale acestei platforme sunt:

- Funcțiile sunt salvate într-o bază de date, având acces la ele de oriunde, fără a avea numeroase fișiere ce ar trebui păstrate, sincronizate, cu back-up-uri etc.;

- Se poate programa ușor, fiind la fel utilizat atât pe desktop, telefon sau tabletă;

- Permite adăugarea unor instanțe noi de pe telefon (lucru ce nu se poate realiza ușor cu alte programe);

- Se pot adăuga oricâte funcții publice și clase noi;

- Posibilitatea folosirii funcțiilor făcute de către alți utilizatori direct de pe platforma Somon (fără a fi nevoie de a lua o bucată de cod de pe internet, cu eventuale dependențe, posibil în alt limbaj de programare și fără nevoia de a fi integrată bucata respectivă cu restul proiectului);

- În afară de Miner nu trebuie instalat nimic (și acesta este opțional dacă sunt lăsați alți utilizatori să mineze);

- Este gratuit;

- Este open-source.

11. Surse folosite

- [1] Dennis J.B. (1974) First version of a data flow procedure language. In: Robinet B. (eds) Programming Symposium. Lecture Notes in Computer Science, vol 19. Springer, Berlin, Heidelberg
- [2] <http://www.ni.com/ro-ro/shop/labview.html>
- [3] <http://jsfiddle.net/tubededentifrice/u5y15d0L/2/>
- [4] <https://github.com/BYVoid/MidiToSheetMusic>
- [5] <https://ffmpeg.org/>
- [6] <http://waon.sourceforge.net/>
- [7] [https://en.wikipedia.org/wiki/Snippet_\(programming\)](https://en.wikipedia.org/wiki/Snippet_(programming))
- [8] <https://www.youtube.com/watch?v=DiMqcXd2nfo>
- [9] <https://github.com/TheAlgorithms>
- [10] https://en.wikipedia.org/wiki/History_of_programming_languages
- [11] <https://github.com/adishavit/argh>
- [12] <http://www.cnet.ro/2012/07/16/scheme-logice-introducere-ii/>
- [13] <http://www.infovalutar.ro/bnr/istoric/eur>
- [14] <http://appinventor.mit.edu/explore/>
- [15] <http://www.flowgorithm.org>
- [16] <http://somon.xyz/>
- [17] https://en.wikipedia.org/wiki/Color_histogram
- [18] Radu Tudor Ionescu, Marius Popescu, Aoife Cahill. String Kernels for Native Language Identification: Insights from Behind the Curtains in MIT Press Journals, 2016 - <https://www.aclweb.org/anthology/J16-3005>
- [19] <http://string-kernels.herokuapp.com/>
- [20] <https://profs.info.uaic.ro/~ciortuz/SLIDES/wsd.pdf>
- [21] https://www.researchgate.net/publication/228998305_Word_Sense_Disambiguation_Using_Word_Ontology_and_Concept_Distribution
- [22] <https://github.com/teo2mirce/Word-Sense-Disambiguation>
- [23] <https://www.google.com/>
- [24] <https://sourceforge.net/projects/archivconvert/>
- [25] <https://www.audacityteam.org/>
- [26] https://www.youtube.com/watch?v=Oe7Ps_hY_Ac
- [27] https://www.youtube.com/watch?v=hP_8E80eW8
- [28] <https://cloud.google.com/vision/>
- [29] <https://github.com/LazoCoder/Article-Summarizer>
- [30] <http://autosummarizer.com/index.php>
- [31] <http://miluzzo.info/pubs/crowdpp.pdf>
- [32] <https://www.pexels.com/photo/agriculture-clouds-countryside-cropland-440731/>