TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Johan Erik Pukk

N-graafide analüsaatori loomine

Bakalaureusetöö (9 EAP)

Juhendajad: Heidi Meier, MSc
Marina Lepp, PhD

Tartu 2021

N-graafide analüsaatori loomine

Lühikokkuvõte:

Programmeerimiskursustel kasutatakse tihti õppimiseks mõeldud programmeerimis-keskkondi, mis koguvad logifailidesse informatsiooni õpilaste tööprotsessi kohta. Logifailide analüüs võib aidata aru saada, kuidas õpilased õpivad, ja üks viis, kuidas logifaile analüüsida, on uurida seal olevaid n-graafe. Selle bakalaureusetöö raames loodi tööriist Thonny logifailidest n-graafide leidmiseks ja esialgseks analüüsiks. Tööriist suudab Thonny logifailidest leida kõik n-graafid vastavalt kasutaja määratud filtritele ja salvestada leitud n-graafid kas CSV- või XLSX-faili.

Võtmesõnad:

N-graafid, Thonny, logifail, programmeerimise õpetamine

CERCS: P175 Informaatika, süsteemiteooria, S281 Arvuti õpiprogrammide kasutamise metoodika ja pedagoogika

Creating an n-graph analyser

Abstract:

Programming courses often use a teaching programming environment that collects information about the students work process into log files. Analysing log files can give insights into how students learn and one method of analysing log files is to study the n-graphs in them. As part of this thesis a tool for parsing n-graphs from Thonnys log files was created. The tool can find all n-graphs in Thonny log files according to user defined filters and save the resulting n-graphs to either a CSV- or XLSX-file.

Keywords:

N-graphs, Thonny, log file, teaching programming

CERCS: P175 Informatics, systems theory, S281 Computer-assisted education

Sisukord

1. Sissejuhatus

Tartu Ülikool pakub mitmeid algajatele suunatud programmeerimiskursuseid, kus osaleb järjest rohkem õppijaid. Need kursused on põhiliselt programmeerimiskeeles Python ja Tartu Ülikoolis kasutatakse Pythoni õpetamiseks programmeerimiskeskkonda Thonny. Thonny loob lahendamise ajal logifaili, kuhu salvestatakse erinevad andmed õppija tööprotsessi kohta. Õppijate tööprotsessi uurides saab leida, kas ülesanded on paraja raskusega ning leida kohti, kus õpilastel tekivad raskused, ja mida peaks kursusel põhjalikumalt käsitlema. Logifailides sisalduv info võib aidata kaasa plagiaadi tuvastamisele.

Logifailis sisalduvate andmete hulka kuulub ka info õpilase klahvivajutuste ja nende vahelise aja kohta. N-graaf on n tähemärgi kombinatsioon, näiteks „f:“ (n = 2) või „i +=“ (n = 4). N-graafide analüüsist on võimalik saada palju infot kirjutaja kohta. Näiteks Leinonen jt [1] leidsid, et programmeerimiskeeltele omaste digraafide kirjutamiseks kulunud aeg on korrelatsioonis õpilaste varasema programmeerimiskogemuse ja eksamitulemusega. N-graafide analüüsi kaudu on ka võimalik programmeerijaid nende koodi kirjutamise viisi põhjal eristada teistest programmeerijatest [2]. Kui on programmeerijaid võimalik üksteisest eristada, siis on n-graafide analüüsil potentsiaali ka plagiaadi tuvastamisel.

N-graafide leidmine logifailist on käsitsi aeglane protsess. Kuna kursustel on palju osalejaid, on ka loodud logifaile palju. Selleks et tulevikus oleks analüüs kiirem ja mugavam, oleks hea, kui saaks digraafide leidmise ja esialgse analüüsi automatiseerida.

Selle bakalaureusetöö eesmärk on luua programm, millega saab automaatselt leida Thonny logifailidest n-graafe vastavalt kasutaja soovidele, neid näidata graafilises kasutajaliideses ja salvestada faili, et oleks võimalik neid teiste tööriistadega uurida. Tarkvara kasutajateks on programmeerimise ainete õppejõud, kes soovivad uurida läbi n-graafide analüüsi oma õpilaste tööprotsessi.

2. N-graafid

Selles peatükis antakse ülevaade n-graafidest, nende tippimisega seotud infost ja varasematest töödest, mis on n-graafe programmeerimise õppimise ja plagiaadituvastuse kontekstis uurinud. N-graafid on n tähe järjestatud kombinatsioon, digraaf on kahe tähe kombinatsioon [1, 2]. Põhiliselt on uuritud digraafe ja nende tippimisega seotud ajavahemikke [1, 2]. Näiteks keskmine aeg esimese klahvivajutuse algusest teise klahvivajutuseni või keskmine aeg enne järgmist klahvivajutust peale mingi kindla digraafi kirjutamist.

Käsikirjas on igal inimesel isesugune käekiri ja sarnaselt on ka igal inimesel erinev tippimisstiil. On ka uuritud, kuidas ning kui täpselt saaks eristada inimesi nende tippimisstiili järgi. N-graafide kasulikkust tehisnärvivõrgu sisendtunnusena on palju uuritud [3, 4]. Lin [3] uuris, kas sisselogimisel saaks lisaks parooli kontrollimisele ka parooli kirjutusstiili tehisnärvivõrgu abil kontrollida. Lin kasutas närvivõrgu sisendiks parooli kirjutamiseks tehtud klahvivajutuste vahelisi aegu ja üksikute klahvide all hoidmise aega. Cho jt [4] uurisid paroolile lisaks tehisnärvivõrguga kirjutamisstiili kontrollimist ja saavutasid sarnase täpsuse, nende keskmine veamäär oli 1%. Paroolide kirjutusstiili kontrollimisel on suur piirang fakt, et paroolid on tavaliselt suhteliselt lühikesed ja seega on andmeid, mille põhjal mudelit treenida ja ennustada, vähe.

Vähesed andmed ei ole probleemiks plagiaadituvastusel programmeerimiskursustel. Eksami või kontrolltöö sooritaja töö käigus kirjutatud n-graafe saab võrrelda selle õppija kursuse käigus tehtud töödega. Longi jt [2] töös uuriti, kui suure täpsusega on võimalik programmeerijaid ära tunda nende kirjutatud digraafide põhjal ja kas programmeerijad on endiselt äratuntavad, kui testandmed korjati teiselt programmeerimiskursuselt, mis toimus hiljem. Samuti võrdlesid nad keskmist aega klahvivajutuste vahel ja kindlate digraafide tippimise kiirust. Nad leidsid, et digraafid andsid õppijate tuvastamisel parema tulemuse ja nende mudeli täpsus oli 95,4%, kui treeningandmeteks olid kursuse 7 nädala tööd. Täpsus langes 77,8% peale, kui treeningandmeteks olid ainult esimese 2 nädala andmed. Seega on n-graafidel potentsiaali plagiaadikontrolli vahendina.

Info programmeerimiskursusel õppijate varasema kogemuse kohta on kasulik kursuste läbiviijatele, sest selle teadmisega saab õppejõud anda paremat tagasisidet ja suunata õppijat paremini. N-graafide kasulikkust on ka selles vallas uuritud. Leinoneni jt [1] töös uuriti, kas on võimalik eristada täiesti algajaid programmeerijaid nendest, kellel oli vähemalt mingisugune varasem kogemus, ja prooviti ka ennustada, kas õppija eksamitulemus on mediaantulemusest

parem või kehvem. Ülesande lahendamiseks kasutasid nad erinevaid masinõppe algoritme ja saavutasid lõpuks täpsuse 71,7% eksamitulemuste ennustamisel ja 74,56% programmeerimiskogemuse ennustamisel.

Edwards jt [5] võrdlesid tudengite tippimisstiili loomuliku keele kirjutamisel ja programmeerimisel. Nende eesmärk oli uurida, kas ja kuidas tudengite tippimisstiilid erinevad sõltuvalt kontekstist ja kuidas tippimisstiil muutub, kui programmeerimisoskus kasvab. Uuriti nelja erinevat konteksti: essee inglise keeles, essee soome keeles, programm Pythonis ja programm Javas. Uurimus toimus kahes ülikoolis, uuriti iganädalaste programmeerimiskursuste ülesannete kirjutamist ja kursuse lõpus kirjutatud esseed. Nad leidsid, et loomulikus keeles kirjutasid tudengid keskmiselt kiiremini, kuid kiiruste vahe vähenes kursuse käigus, kuna programmeerimiskeeles kirjutamise kiirus oli kursuse viimasel nädalal kiirem kui esimesel nädalal. Nad leidsid veel, et kindlate digraafide tippimise kiirus sõltus oluliselt ka sõna-kontekstist, näiteks digraaf „nt“ kirjutati oluliselt kiiremini sõnas „print“ kui sõnas „int“. Sõna-konteksti olulisuse võimalikuks põhjuseks toodi välja, et „int()“ funktsiooni õpetati kursusel hiljem kui „print()“ ja selle tõttu oli tudengitel vähem aega sellega tutvuda. Teine välja toodud võimalik põhjus on digraafi asukoht sõnas, kuna „print“ on pikem sõna ja „nt“ esineb lõpus, on kirjutajal juba hoog sees, kui ta kirjutab viimased kaks tähte. Samas töös uuriti ka, kui tihti tudengid kirjutades vigu teevad ja kui kiirelt nad tehtud vigu parandavad. Pythonis tehtud vigade parandamise kiirus paranes kursuse lõpu poole, kuid Javas tehtud vigade parandamise kiirus püsis kursuse algusest lõpuni sama. Võimaliku põhjendusena toodi välja asjaolu, et soome keel ei sarnane Java koodiga, aga inglise keel sarnaneb Pythoni koodiga. Vigade parandamise kiirus Pythonis küll suurenes, aga väikesel määral, seda põhjendati programmeerimiskursuse ülesehitusega. Kursuse käigus õpetati tudengitele järjest uusi keelekonstruktsioone ja seetõttu oli neil kogu kursuse vältel võimalik uusi vigu teha. Vigade parandamise kiirus oli loomulikus keeles kiirem kui programmeerimiskeeles isegi kursuse lõpus.

Byun jt [6] uurisid kuidas n-graafide põhjal tuvastada plagiaati algajatele suunatud programmeerimiskursustel. Tavalised meetodid koodiplagiaadi tuvastamiseks nagu abstraktsete süntaksipuude võrdlus või koodi stiili analüüsivad algoritmid ei tööta algajate kirjutatud koodil hästi, sest algajatele antud ülesanded on tüüpiliselt lühikeste lahendustega. Selliseid lihtsaid ülesandeid lahendatakse tihti sarnaselt ja selle tõttu leiavad tavalised plagiaadituvastusalgoritmid palju valepositiivseid tulemusi. Byun jt treenisid otsustusmetsa, millega nad proovisid tuvastada, kas tudeng oli esitatud programmi ise kirjutanud. Andmeteks

olid kursuse „Sissejuhatus programmeerimisse“ käigus 27 tudengi lahendatud 5 programmeerimisülesannet. Plagiaati simuleeriti ühe tudengi ülesande lahenduse asendamisega teise tudengi lahendusega. Teise tudengi andmed eemaldati ka treeningandmetest, et simuleerida olukorda, kus tudeng esitab kolmanda isiku poolt kirjutatud lahenduse. Üle kõikide tudengite ja ülesannete kombinatsioonide saadi tulemuseks 85,9% täpsus (ingl *accuracy*) ja 84,4% saagis (ingl *recall*, statistikas tõesete positiivsete arv jagatud kõikide positiivsete arvuga).

3. Python ja Thonny

3.1 Python

Tartu Ülikooli algajatele suunatud programmeerimiskursused õpetavad enamasti programmeerimiskeelt Python. Python on kõrgetasemeline programmeerimiskeel, millega saab luua nii suuri süsteeme kui ka lihtsaid skripte. Python on algajatele suunatud kursuste keelevalik just sellepärast, et Python on programmeerimiskeel, mis on algajatele arusaadavam kui mõned teised keeled [7]. Suur osa Pythoni loetavusest tuleb sellest, et Pythoni kood näeb tihti välja peaaegu nagu tavaline inglise keel. Näiteks järgnev koodijupp (joonis 1):

```
if x is not None and len(x) == y:
    print(x)
```

Joonis 1. Näide Pythoni koodist

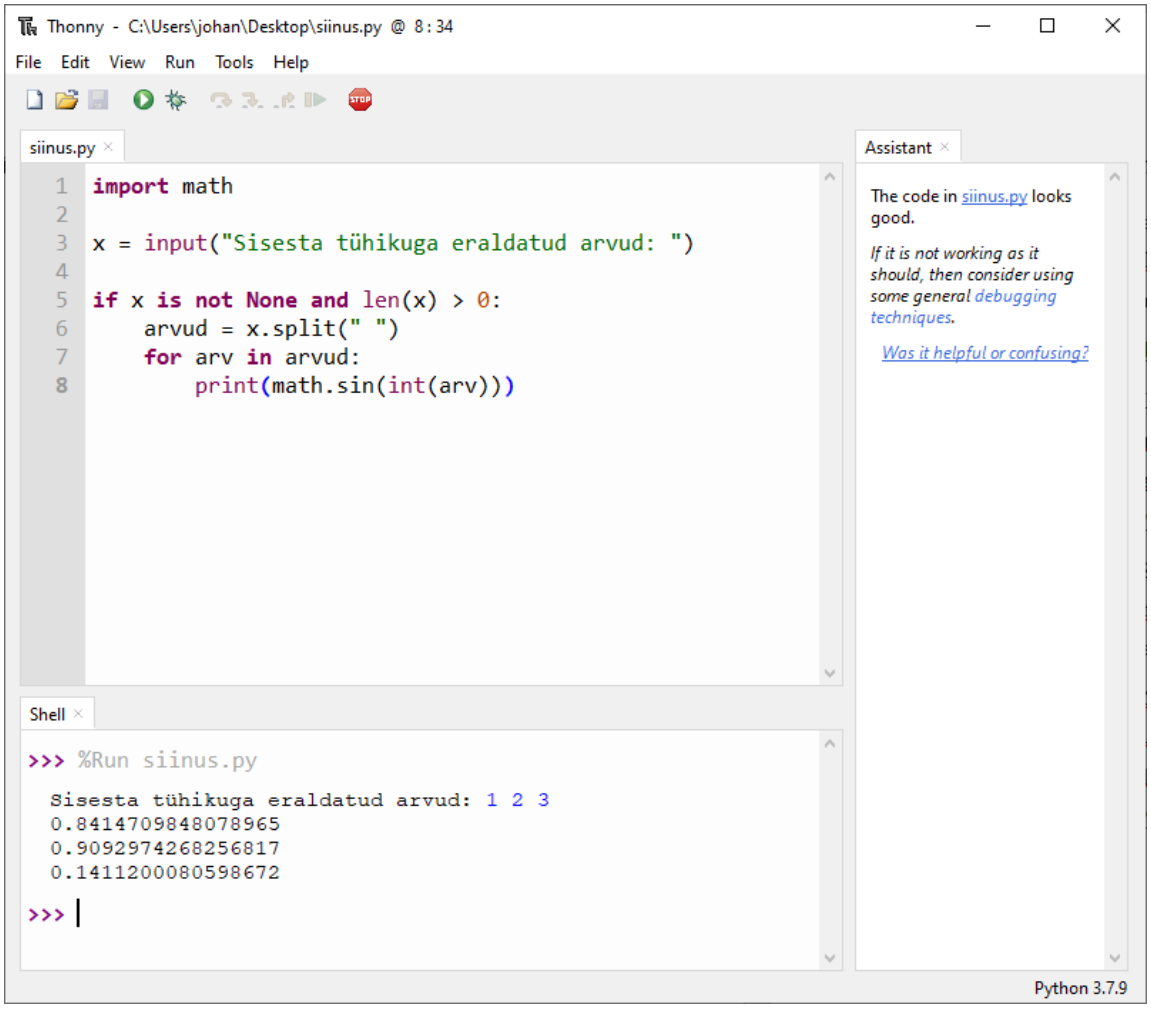
See kood kontrollib, kas muutuja x on tühi ja kas muutuja x pikkus on võrdne muutuja y väärtusega. Kui mõlemad tingimused on tõesed, siis väljastatakse muutuja x väärtus konsooli.

3.2 Programmeerimiskeskkond Thonny

Thonny on Tartu Ülikoolis loodud Pythoni arenduskeskkond, mis on mõeldud programmeerimise õppimiseks [8]. Thonny on Tartu Ülikooli programmeerimise algkursuste eelistatud arenduskeskkond. Thonny on hea esimene arenduskeskkond, sest Thonnyl on lihtsustatud ja intuitiivne kasutajaliides. Professionaalsetel arenduskeskkondadel on väga palju erinevaid valikuid ja funktsionaalsust, mida algajatel nagunii vaja ei ole ning pigem ajab kasutaja segadusse. Thonnyl on põhiekraanil koodiredaktor; Pythoni käsurida, kus saab lühikesi koodijuppe kiirelt proovida; ja assistent, mis aitab vältida ning parandada tihti esinevaid vigu (joonis 2).

Üks olulisemaid Thonny funktsioone on kergesti kasutatav silur (ingl *debugger*). Thonny silur astub programmi sammhaaval läbi samas aknas, kus kasutaja programmi kirjutas. Silumise funktsionaalsus on oluline, sest see on väga hea tööriist programmeerijatele vigade leidmiseks ja arusaamiseks, kuidas nende kirjutatud koodi täidetakse.

Üks Thonny funktsionaalsus, mis n-graafe mõjutab, on koodi automaatne lõpetamine (ingl *code completion*), ehk Pythoni võtmesõnade kirjutamisel pakub Thonny poole sõna pealt võimalust automaatselt võtmesõna lõpetada tab-klahviga. Selle tõttu võib logifailides esineda vähem võtmesõnade lõppudega seotud kirjeid.



Joonis 2. Thonny koodiredaktor

3.3 Thonny logifailid

Thonnyl on ka logide funktsionaalsus, mis salvestab kõik õpilase tegevused nii koodiredaktoris kui ka käsuaknas.

Thonny logid on JSON (ingl *JavaScript Object Notation*) vormingus tekstifailid. Thonny logifailide struktuur on väga põhjalikult kirjeldatud Roosi töös [9]. Logis olevad kirjed koosnevad võti-väärtus paaridest, kus võti on alati sõne ja väärtus võib olla mis tahes tüüpi. Käesoleva töö jaoks on olulised need logikirjed, mis kirjeldavad koodiredaktorisse kirjutamist. Joonis 3 on näide sellisest kirjest.

```
{
    "index": "13.0",
    "text": "i",
    "tags": "None",
    "text_widget_id": 87036848,
    "text_widget_class": "CodeViewText",
    "trivial_for_coloring": true,
    "trivial_for_parens": true,
    "sequence": "TextInsert",
    "time": "2020-03-22T21:22:55.058042"
}
```


Joonis 3. Thonny logifaili kirje

See logikirje näitab, et 13. reale, 0. veergu ("index": "13.0") sisestati täht „i“ ("text": "i").

- „tags“ väärtuses võivad olla lisainfot andvad sildid, väärtus „None“ tähendab, et sildid puuduvad.
- „text_widget_id” väärtus näitab, millisesse Thonny aknasse sümbol kirjutati, see on vajalik, kuna Thonnys on võimalik töötada mitmes failis korraga.
- „text_widget_class“ väärtus näitab, mis tüüpi aknaga on tegemist, koodiredaktori puhul alati "CodeViewText".
- "trivial_for_coloring" ja "trivial_for_parens" parameetrid näitavad, kas sisestatud sümbol on koodiredaktoris näidatava värvi ja sulgude lõpetamiseks kerge.
- "sequence" näitab, mis tegevus tehti, selles näites "TextInsert" ehk teksti sisestamine.
- "time" näitab tegevuse toimumise aega.

Thonny logifailide põhikasutus on kasutussessioonide taasesitamise funktsionaalsus, kus on võimalik näha täpselt, mida kasutaja tegi. Selline tähthaaval kordus on kasulik õppejõududele, et uurida, kus tudengid kauem mõtlesid või proovida aru saada nende mõttekäigust. Thonny logide taasesitaja avamiseks tuleb kõigepealt muuta Thonny kasutajaliidese režiimi. Selleks tuleb valida menüüst *Tööriistad -> Seaded...* ja avanevast menüüst leida *Kasutajaliidese režiim* ja valida rippmenüüst *expert*. Kui see on tehtud, siis tuleb Thonny taaskäivitada ja valides menüüst *Tööriistad -> Ava taasesitaja…,* avaneb nüüd Thonny taasesitaja.

4. N-graafide analüsaator

Selles peatükis antakse ülevaade töö raames loodud programmist, selle loomisprotsessist ja kasutamisest.

4.1 Programmi loomine

Programm on kirjutatud Pythonis. Arenduskeskkonnaks oli JetBrainsi PyCharm Professional ja kasutusel oli versioonihaldus Git, programmi lähtekood on saadaval avalikult Githubis (lisa 1). Programmeerimiskeele valikul on mitu põhjust. Esiteks on Pythonis väga suur moodulite valik, töö kirjutamise ajal oli Pythoni moodulite indeksis 297000 projekti [10]. Suur moodulite valik on kasulik, sest tihti leidub moodul, mis oluliselt lihtsustab ja kiirendab programmi arendust. Teine suur põhjus on suur Pythoni kompetents Tartu Ülikoolis. Kuna programmi kasutajate sihtgrupiks on Tartu Ülikooli Pythoni kursuste õppejõud, siis on võimalus programmi kasutajatel vajadusel tulevikus programmile funktsionaalsust lisada. Suureks faktoriks programmeerimiskeele valikul oli ka autori mugavus Pythonis.

Programmis on kasutatud mitut Pythoni moodulit. Peamised neist on Pandas, Matplotlib ja Pandastable. Programmi tööks on vajalikud ka mõned teised moodulid, aga neid siin eraldi välja ei tooda, sest neid on põhiliselt vaja 3 peamise mooduli tööks ja programm neid ise ei kasuta. Kõik moodulid on kirjas programmi kaustas *requirements.txt* failis, mille abil *install.py* need automaatselt paigaldab.

Pandas on andmeteaduse moodul, mille abil saab efektiivselt hoida ja töödelda suuri andmehulki [11]. DataFrame on klass moodulis Pandas, milles saab hoida erinevat tüüpi andmeid, samuti annab see klass mitmeid erinevaid meetodeid andmete töötlemiseks. Selles töös kasutatakse Pandase DataFrame'i logifalidest parsitud andmete hoidmiseks.

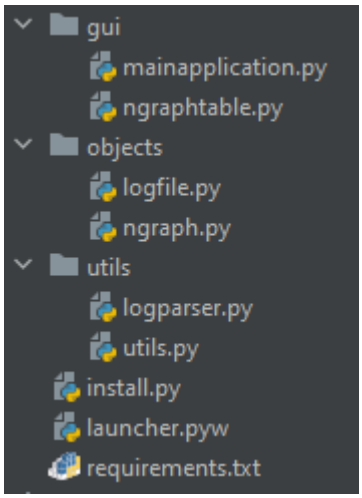
Matplotlib on 2D graafika moodul, mille abil saab kergelt visualiseerida andmeid [12]. Matplotlib on väga võimas tööriist, millega on võimalik ehitada erinevat tüüpi graafikuid ja jooniseid. Selles töös kasutatakse Matplotlib moodulit lihtsate tulpdiagrammide näitamiseks.

Pandastable on moodul, mille abil saab Pandase DataFrame'i mugavalt integreerida Tkinteri abil loodud graafilisse kasutajaliidesesse [13]. DataFrame kuvatakse interaktiivse tabelina. Pandastable moodulis on Pythoni klass Table, mida saab kasutada kohe või luua enda alamklass, et teha muudatusi tabeli vaikimisi käitumisele. Selles töös kasutatakse Pandastable mooduli Table klassi ülemklassina ja loodud alamklass on kasutajaliidese osa, mis kuvab logifailist leitud n-graafid tabelis. Alamklassi muudatused on väikesed, kuid

kasutajasõbralikkuse poole pealt olulised. Nimelt on eemaldatud hüpikaknad, mis küsivad kasutajalt kinnitust tabeli muutmise korral, näiteks mõne veeru järgi sorteerimisel. Teine muudatus, mida teeb loodud alamklass, on võimalus salvestada tabeli sisu uuemas Microsoft Exceli vormingus. Vaikimisi suudab Table klass salvestada ainult CSV-, XLS- või PICKLE-vormingus, loodud alamklass lisab nende hulka XLSX-vormingu. XLS-vorming oli kasutusel Microsoft Exceli vaikimisi failivorminguna kuni 2007, hilisemad Exceli versioonid kasutavad vaikimisi XLSX-failivormingut [14]. Samuti muudeti CSV-vormingus salvestamisel andmete eraldaja komast semikooloniks, sest katsetamise käigus tuli välja, et eestikeelne Windows ootab CSV-failidel andmete eraldajaks semikoolonit, mitte koma.

4.2 Programmi struktuur

Programm on kirjutatud kasutades objektorienteeritud programmeerimise põhimõtteid. Programm koosneb kokku kaheksast Pythoni failist kolmes kaustas (joonis 4).



Joonis 4. Programmi kaust Pycharm IDEs

Kaustas nimega *gui* on kasutajaliidese elemendid. Kasutajaliides on loodud Pythoni standardteeki kuuluva Tkinter mooduliga. Failis *mainapplication.py* on graafilise kasutajaliidese juurelement ja kõikide nuppude loogika. Failis *ngraphtable.py* on klass NGraphTable, mis on Pandastable mooduli klassi Table alamklass. Kaustas nimega *objects* on programmi poolt kasutatavad andmestruktuuride klassid. Failis *logfile.py* on klass LogFile, mis modelleerib ühte logifaili. LogFile klassi konstruktor võtab parameetriks logifaili nime ja avalikustab meetodi, millega saab leida sellest failist kõik n-graafid. Failis *ngraph.py* on klass NGraph, mis modelleerib ühte n-graafi ja hoiab kogu selle n-graafiga seotud informatsiooni.

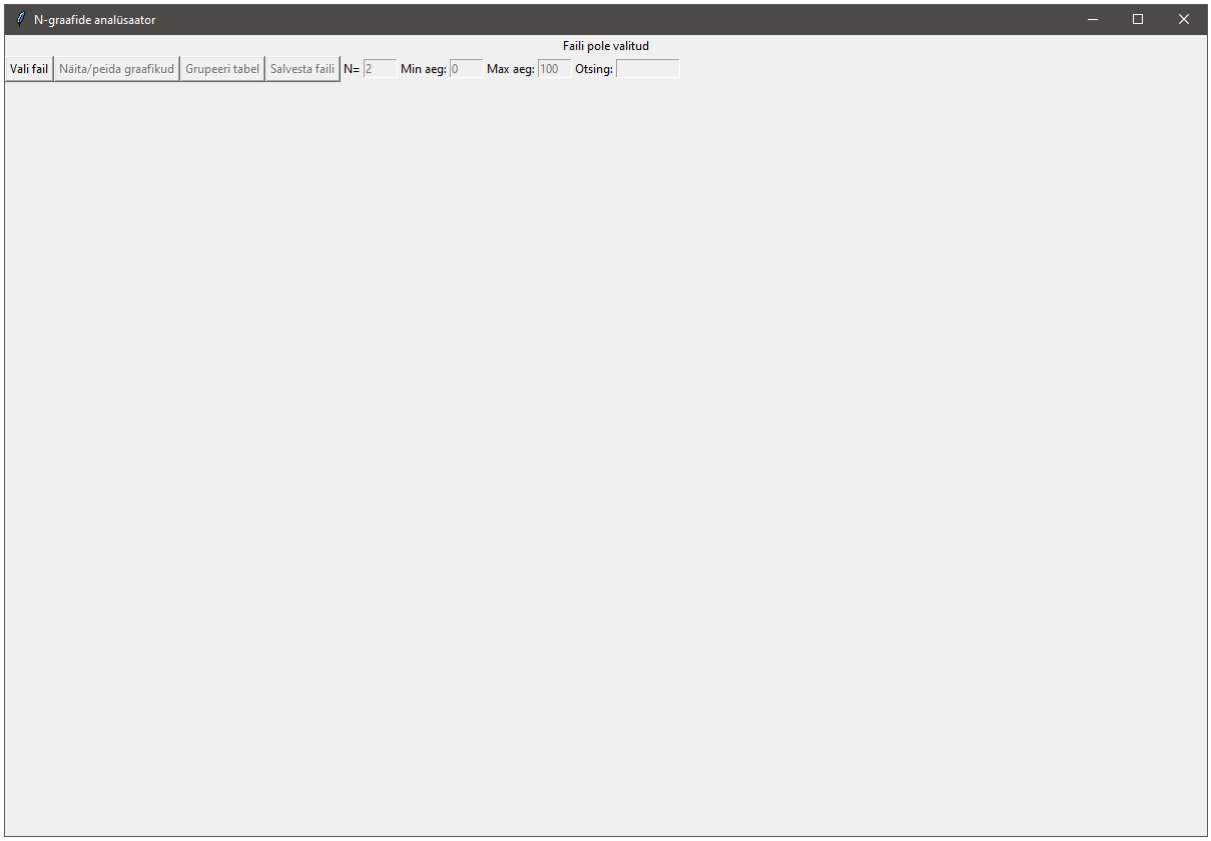
Kaustas *utils* on üldiselt kasulikud funktsioonid, mida kasutab rohkem kui üks klass. Failis *logparser.py* on logifailide lugemisega seotud funktsioonid ja failis *utils.py* muud kasulikud funktsioonid.

Failis *requirements.txt* on kirjas kõik programmi tööks vajalikud moodulid, mis ei ole Pythoni standardteegis. See fail on automaatselt genereeritud järgmise käsu abil: `pip freeze > requirements.txt`

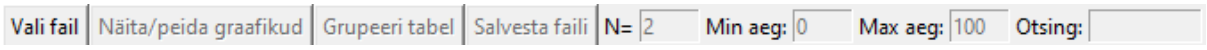
Install.py ja *launcher.pyw* on vastavalt skriptid, mille abil saab vajalikud moodulid paigaldada ja programmi käivitada. *Install.py* kutsub välja Pythoni moodulite paigaldamise tööriista *pip*, andes sellele argumendiks faili *requirements.txt*. *Launcher.pyw* käivitab graafilise kasutajaliidese. Pyw faililaiendiga Pythoni skripte käivitades ei tehta vaikimisi lahti süsteemi terminali (Windowsis *command prompt*).

4.3 Kasutusjuhend

Enne programmi esimest korda kasutamist tuleb esmalt käivitada *install.py*. Kui see on oma töö lõpetanud, võib käivitada *launcher.pyw*, et avada programmi kasutajaliides (joonis 5). Edaspidi võib käivitada otse *launcher.pyw*.

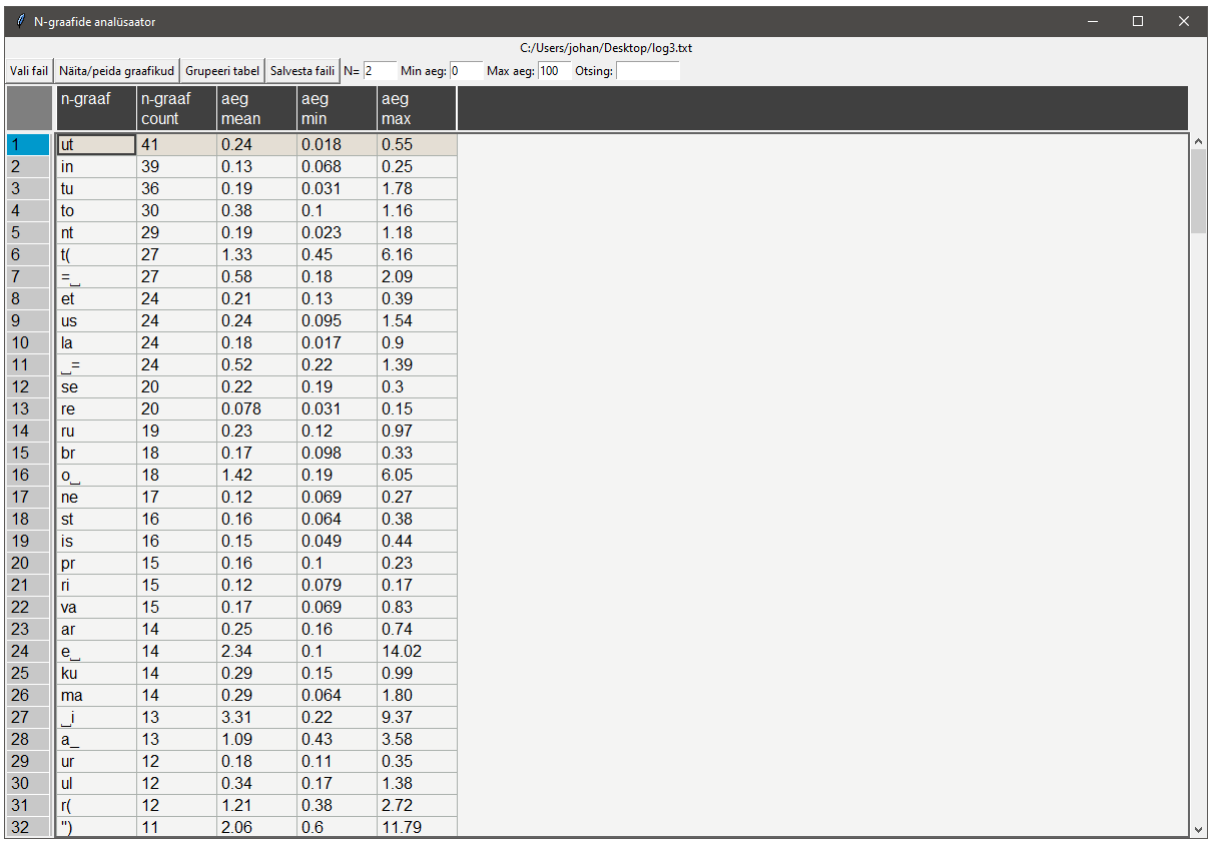


Joonis 5. N-graafide analüsaator vahetult peale käivitamist



Joonis 6. Tööriistariba

Tööriistaribal (joonis 6) on esialgu aktiivne ainult üks nupp (*Vali fail*), millele vajutades saab valida oma arvutist logifaili, mida analüüsida. Nupule vajutades tuleb lahti operatsioonisüsteemi failihaldur, kus tuleb soovitud fail valida. Peale faili valimist parsib programm seda faili ja koostab leitud digraafidest tabeli (joonis 7).



	n-graaf	n-graaf count	aeg mean	aeg min	aeg max
1	ut	41	0.24	0.018	0.55
2	in	39	0.13	0.068	0.25
3	tu	36	0.19	0.031	1.78
4	to	30	0.38	0.1	1.16
5	nt	29	0.19	0.023	1.18
6	t(	27	1.33	0.45	6.16
7	=_	27	0.58	0.18	2.09
8	et	24	0.21	0.13	0.39
9	us	24	0.24	0.095	1.54
10	la	24	0.18	0.017	0.9
11	_=	24	0.52	0.22	1.39
12	se	20	0.22	0.19	0.3
13	re	20	0.078	0.031	0.15
14	ru	19	0.23	0.12	0.97
15	br	18	0.17	0.098	0.33
16	o_	18	1.42	0.19	6.05
17	ne	17	0.12	0.069	0.27
18	st	16	0.16	0.064	0.38
19	is	16	0.15	0.049	0.44
20	pr	15	0.16	0.1	0.23
21	ri	15	0.12	0.079	0.17
22	va	15	0.17	0.069	0.83
23	ar	14	0.25	0.16	0.74
24	e_	14	2.34	0.1	14.02
25	ku	14	0.29	0.15	0.99
26	ma	14	0.29	0.064	1.80
27	_i	13	3.31	0.22	9.37
28	a_	13	1.09	0.43	3.58
29	ur	12	0.18	0.11	0.35
30	ul	12	0.34	0.17	1.38
31	r(	12	1.21	0.38	2.72
32	")	11	2.06	0.6	11.79

Joonis 7. Analüsaator koos avatud failiga

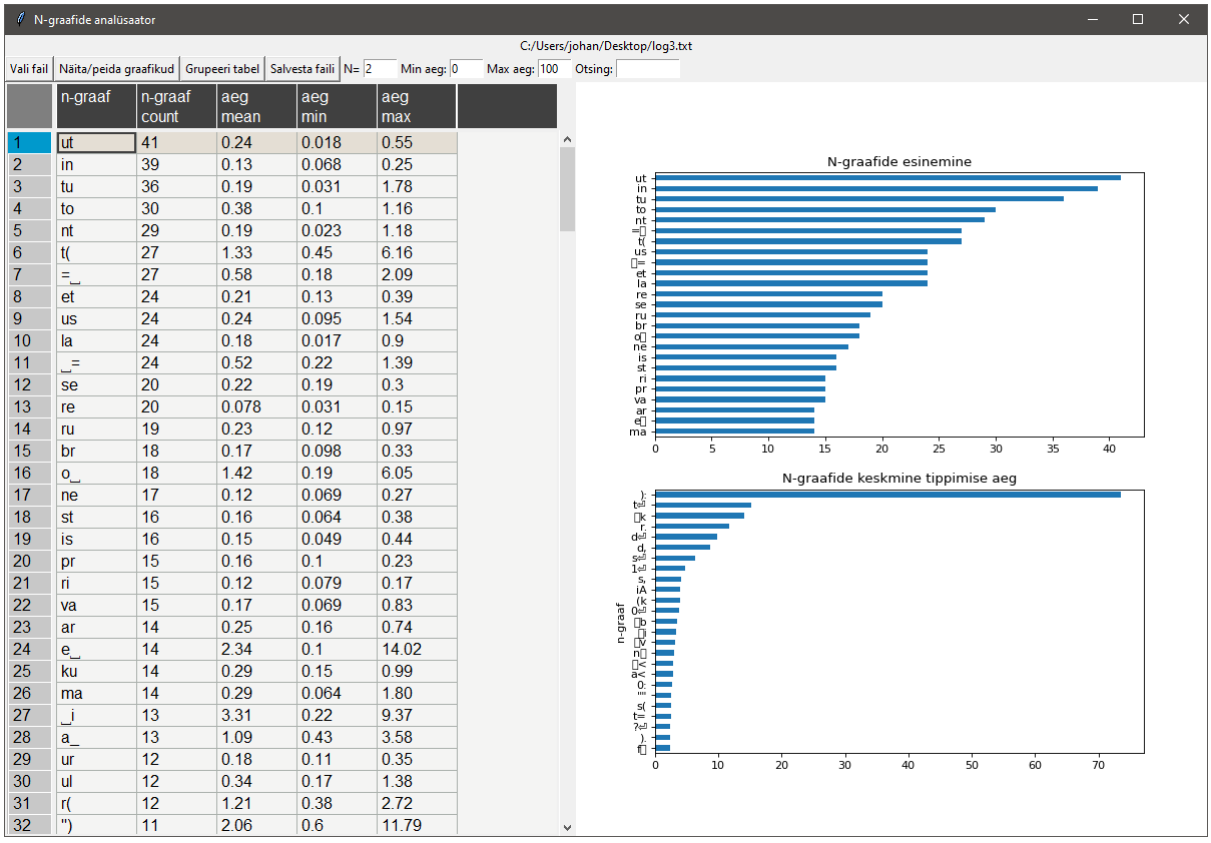
Genereeritud tabelis on veerud *n-graaf, n-graaf count, aeg mean, aeg min* ja *aeg max*. Veergude nimetused on osaliselt inglise keeles, kuna need on genereeritud automaatselt Pandase DataFrame.agg meetodiga. Testimisel tekkis osadel arvutitel probleem, kus tabeli veergude nimetused ei mahtunud veergude sisse ära ja olid seega osaliselt peidetud. Probleem sai küll avastatud, aga kuna defekti ei suudetud kordagi esile kutsuda arenduseks kasutatud arvutites, ei leitud selle defekti põhjust ega ka lahendust.

Veergude tähendused on järgmised:

- n-graaf - tähed, millest n-graaf koosneb;
- n-graaf count - mitu sellist n-graafi leiti;
- aeg mean - keskmine aeg sekundites, mis kulus selle n-graafi kirjutamiseks;
- aeg min - minimaalne aeg sekundites, mis kulus selle n-graafi kirjutamiseks;
- aeg max - maksimaalne aeg sekundites, mis kulus selle n-graafi kirjutamiseks.

Tabelis on veerus *n-graaf* tühikud ja reavahetused asendatud vastavalt ␣ ja ⏎ sümbolitega, et neid oleks lihtsam lugeda.

Peale faili avamist muutuvad teised nupud aktiivseks. Nupp *Näita/peida graafikud* toob esile graafikud, millel on tulpdiagrammidega näidatud tabelis olevate n-graafide esinemine ja keskmine tippimise aeg (joonis 8). Sama nupuga saab ka graafikud uuesti peita. Graafikul y-teljel olevad ruudud on üks probleemidest, millele arenduse käigus lahendust ei leidnud. Kuna tabelis on raskesti loetavad sümbolid asendatud, siis proovivad ka graafikud ␣ sümbolit näidata, aga Matplotlibi sisseehitatud fondis ei ole ␣ sümbolit ja seda näidatakse ruuduga.



	n-graaf	n-graaf count	aeg mean	aeg min	aeg max
1	ut	41	0.24	0.018	0.55
2	in	39	0.13	0.068	0.25
3	tu	36	0.19	0.031	1.78
4	to	30	0.38	0.1	1.16
5	nt	29	0.19	0.023	1.18
6	t(	27	1.33	0.45	6.16
7	=␣	27	0.58	0.18	2.09
8	et	24	0.21	0.13	0.39
9	us	24	0.24	0.095	1.54
10	la	24	0.18	0.017	0.9
11	␣=	24	0.52	0.22	1.39
12	se	20	0.22	0.19	0.3
13	re	20	0.078	0.031	0.15
14	ru	19	0.23	0.12	0.97
15	br	18	0.17	0.098	0.33
16	o␣	18	1.42	0.19	6.05
17	ne	17	0.12	0.069	0.27
18	st	16	0.16	0.064	0.38
19	is	16	0.15	0.049	0.44
20	pr	15	0.16	0.1	0.23
21	ri	15	0.12	0.079	0.17
22	va	15	0.17	0.069	0.83
23	ar	14	0.25	0.16	0.74
24	e␣	14	2.34	0.1	14.02
25	ku	14	0.29	0.15	0.99
26	ma	14	0.29	0.064	1.80
27	␣i	13	3.31	0.22	9.37
28	a␣	13	1.09	0.43	3.58
29	ur	12	0.18	0.11	0.35
30	ul	12	0.34	0.17	1.38
31	r(	12	1.21	0.38	2.72
32	")	11	2.06	0.6	11.79

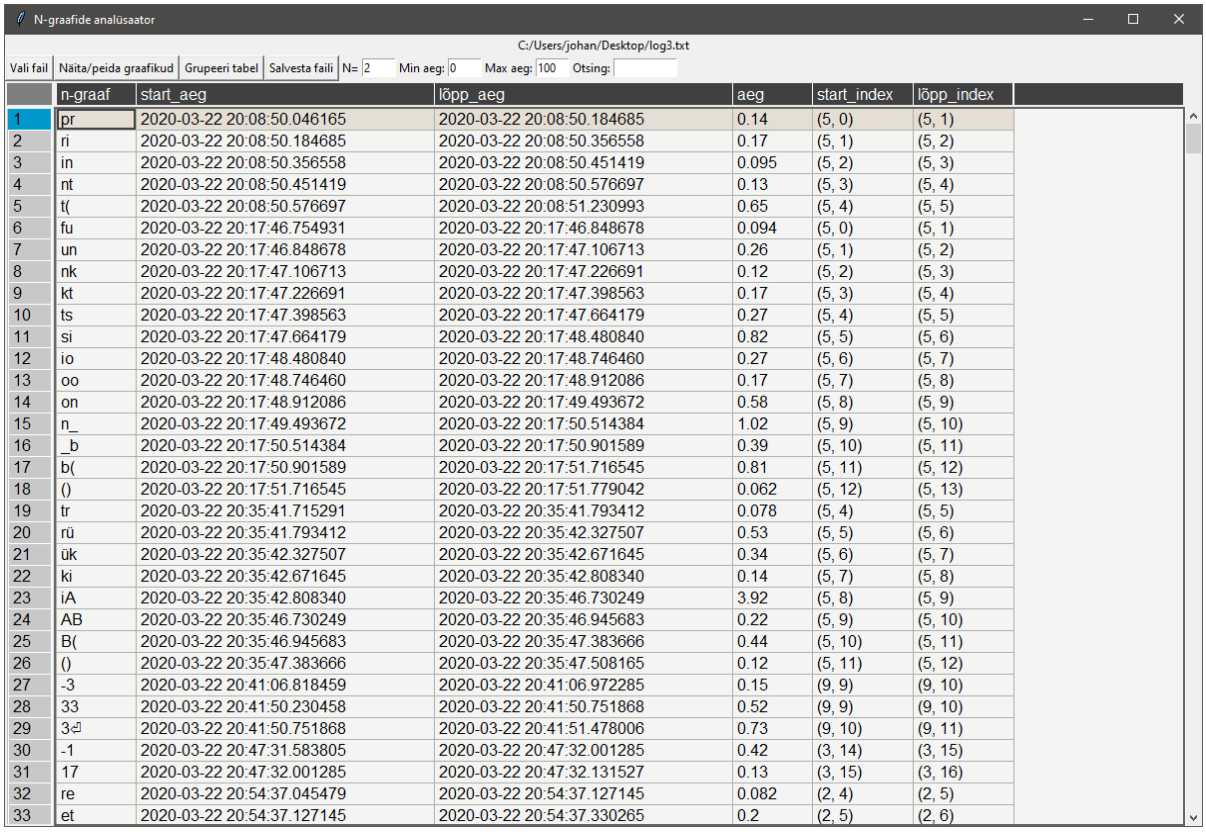
Joonis 8. Analüsaator koos graafikutega

Nupp *Salvesta faili* annab kasutajale võimaluse salvestada tabelis olevad andmed kas CSV- või XLSX-faili. Kui plaan on andmeid edasi töödelda Microsoft Excelis, siis on soovitatav salvestada XLSX-vormingus, kuna CSV-failide Excelis avamine ei pruugi kohe töötada, sest Excel eeldab CSV-failidele erinevat vormingut olenevalt kasutaja Windowsi keele seadetest. Seega ei saa kindel olla, et sama CSV-fail avaneb erinevatel kasutajatel sama moodi. XLSX-failina salvestades tuleb praeguse versiooni puhul arvestada sellega, et võrdusmärgiga algavad tabeli read tõlgendab Excel valemina ja loeb need read valesti sisse.

Tööriistaribal on ka järgmised tabeli filtreerimisvalikud:

- N - muudab n väärtuse n-graafides, vaikimisi 2.
- Min aeg - minimaalne n-graafi tippimiseks kulunud aeg sekundites, mida tabelis kuvada ja arvutustes kasutada, vaikimisi 0.
- Max aeg - maksimaalne n-graafi tippimiseks kulunud aeg sekundites, mida tabelis kuvada ja arvutustes kasutada, vaikimisi 100.
- Otsing - filtreerib tabelit nii, et tabelis oleks ainult need n-graafid, mis sisaldavad selle väärtust, tühja puhul filtreerimist ei toimu.

Nupp *Grupeeri tabel* vahetab grupeeritud ja mittegrupeeritud tabeli vahel (joonis 9), programm alustab grupeeritud tabeliga. Grupeeritud tabelis on kõik samad n-graafid võetud kokku üheks reaks. Programmi jaoks on kaks n-graafi samad, kui need koosnevad samadest tähtedest samas järjekorras.



	n-graaf	start_aeg	lõpp_aeg	aeg	start_index	lõpp_index
1	pr	2020-03-22 20:08:50.046165	2020-03-22 20:08:50.184685	0.14	(5, 0)	(5, 1)
2	ri	2020-03-22 20:08:50.184685	2020-03-22 20:08:50.356558	0.17	(5, 1)	(5, 2)
3	in	2020-03-22 20:08:50.356558	2020-03-22 20:08:50.451419	0.095	(5, 2)	(5, 3)
4	nt	2020-03-22 20:08:50.451419	2020-03-22 20:08:50.576697	0.13	(5, 3)	(5, 4)
5	t(	2020-03-22 20:08:50.576697	2020-03-22 20:08:51.230993	0.65	(5, 4)	(5, 5)
6	fu	2020-03-22 20:17:46.754931	2020-03-22 20:17:46.848678	0.094	(5, 0)	(5, 1)
7	un	2020-03-22 20:17:46.848678	2020-03-22 20:17:47.106713	0.26	(5, 1)	(5, 2)
8	nk	2020-03-22 20:17:47.106713	2020-03-22 20:17:47.226691	0.12	(5, 2)	(5, 3)
9	kt	2020-03-22 20:17:47.226691	2020-03-22 20:17:47.398563	0.17	(5, 3)	(5, 4)
10	ts	2020-03-22 20:17:47.398563	2020-03-22 20:17:47.664179	0.27	(5, 4)	(5, 5)
11	si	2020-03-22 20:17:47.664179	2020-03-22 20:17:48.480840	0.82	(5, 5)	(5, 6)
12	io	2020-03-22 20:17:48.480840	2020-03-22 20:17:48.746460	0.27	(5, 6)	(5, 7)
13	oo	2020-03-22 20:17:48.746460	2020-03-22 20:17:48.912086	0.17	(5, 7)	(5, 8)
14	on	2020-03-22 20:17:48.912086	2020-03-22 20:17:49.493672	0.58	(5, 8)	(5, 9)
15	n_	2020-03-22 20:17:49.493672	2020-03-22 20:17:50.514384	1.02	(5, 9)	(5, 10)
16	_b	2020-03-22 20:17:50.514384	2020-03-22 20:17:50.901589	0.39	(5, 10)	(5, 11)
17	b(	2020-03-22 20:17:50.901589	2020-03-22 20:17:51.716545	0.81	(5, 11)	(5, 12)
18	()	2020-03-22 20:17:51.716545	2020-03-22 20:17:51.779042	0.062	(5, 12)	(5, 13)
19	tr	2020-03-22 20:35:41.715291	2020-03-22 20:35:41.793412	0.078	(5, 4)	(5, 5)
20	rü	2020-03-22 20:35:41.793412	2020-03-22 20:35:42.327507	0.53	(5, 5)	(5, 6)
21	ük	2020-03-22 20:35:42.327507	2020-03-22 20:35:42.671645	0.34	(5, 6)	(5, 7)
22	ki	2020-03-22 20:35:42.671645	2020-03-22 20:35:42.808340	0.14	(5, 7)	(5, 8)
23	iA	2020-03-22 20:35:42.808340	2020-03-22 20:35:46.730249	3.92	(5, 8)	(5, 9)
24	AB	2020-03-22 20:35:46.730249	2020-03-22 20:35:46.945683	0.22	(5, 9)	(5, 10)
25	B(	2020-03-22 20:35:46.945683	2020-03-22 20:35:47.383666	0.44	(5, 10)	(5, 11)
26	()	2020-03-22 20:35:47.383666	2020-03-22 20:35:47.508165	0.12	(5, 11)	(5, 12)
27	-3	2020-03-22 20:41:06.818459	2020-03-22 20:41:06.972285	0.15	(9, 9)	(9, 10)
28	33	2020-03-22 20:41:50.230458	2020-03-22 20:41:50.751868	0.52	(9, 9)	(9, 10)
29	3⏎	2020-03-22 20:41:50.751868	2020-03-22 20:41:51.478006	0.73	(9, 10)	(9, 11)
30	-1	2020-03-22 20:47:31.583805	2020-03-22 20:47:32.001285	0.42	(3, 14)	(3, 15)
31	17	2020-03-22 20:47:32.001285	2020-03-22 20:47:32.131527	0.13	(3, 15)	(3, 16)
32	re	2020-03-22 20:54:37.045479	2020-03-22 20:54:37.127145	0.082	(2, 4)	(2, 5)
33	et	2020-03-22 20:54:37.127145	2020-03-22 20:54:37.330265	0.2	(2, 5)	(2, 6)

Joonis 9. Mittegrupeeritud tabel

4.4 Edasised võimalused

Selle töö raames koostatud programmi abil saab leida Thonny logifailidest n-graafe, uurida leitud n-graafide tippimise aja statistikat ning esinemist ja visualiseerida neid andmeid graafikute abil, aga võimalusi programmi täiendamiseks on mitmeid. Loodud programm suudab analüüsida ühte logifaili korraga, kuid kasulik oleks ka mitme logifaili korraga uurimine, näiteks andes programmile töötlemiseks kausta või ZIP-faili. Mitme logifaili korraga töötlemine oleks kasulik, kui on vaja uurida näiteks ühe õppija terve kursuse töö käigus loodud logifaile või uurida korraga mitme õppija töö käigus loodud logifaile.

Teine võimalus oleks suurem granulaarsus andmetes, kus see võimalik on, näiteks trigraafide (n=3) puhul kuvada ka andmeid igasse trigraafi kuuluvate digraafide (n=2) kohta. Selline funktsionaalsus vähendaks töötlust, mida peaks tegema teistes tööriistades ja seega kiirendaks uurimist.

Praegu suudab programm kuvada ühe n väärtusega n-graafe korraga, potensiaalselt oleks kasulik ka kuvada mitu n väärtust korraga, et saaks neid võrrelda ja ühte faili salvestada

lihtsamaks edasiseks töötlemiseks. Selle tööga valminud tööriistas see võimalik ei ole, aga selle funktsionaalsuse lisamine ei vaja suuri disaini ega *back-end* muudatusi.

Tööriistas kuvatavatel graafikutel on hetkel ␣ asendatud □ sümboliga, kuna Matplotlibi sisseehitatud fondis puudub sümbol ␣ ning kõik sümbolid, mis fondist puuduvad, kuvatakse valge ruuduga. Võimalik lahendus sellele probleemile oleks leida vabavaraline font, milles on olemas kõik vajalikud sümbolid, ja panna see programmiga kaasa ning installida font *install.py* töö käigus. Töö kirjutamise ajal ei õnnestunud leida sobivat fonti.

Tabeli veergude nimetused on hetkel osaliselt ingliskeelsed, kuna osad veerud on genereeritud automaatselt. Võimalik edasine arendus on need eraldi muuta eestikeelseks.

5. Kokkuvõte

Selle bakalaureusetöö eesmärk oli luua tööriist, millega saab Thonny logifailidest leida n-graafe, neid uurida graafilises kasutajaliideses ja salvestada faili, et andmeid edasi töödelda muude tööriistadega. Loodud tööriist suudab töödelda Thonny logifaile, leida neist kõik n-graafid vastavalt kasutaja määratud filtritele ja salvestada leitud n-graafid kas CSV- või XLSX-faili. Tööriist on loodud Pythonis, kasutades mitmeid vabavaralisi mooduleid, sealhulgas Pandas, Matplotlib ja Pandastable. Graafiline kasutajaliides on loodud Pythoni standardteeki kuuluvas Tkinteris.

Töös tutvustati n-graafidega seotud teoreetilist tausta ja varem tehtud töid. Tutvustati ka programmeerimiskeelt Python, programmeerimiskeskkonda Thonny ja Thonny logide süsteemi. Samuti tutvustati loodud tarkvaralahenduse struktuuri ja koostati programmile kasutusjuhend. Tööriist täidab esialgseid nõudeid, kuid kindlasti on võimalik seda edasi arendada. Peatükis 4.4 toodi välja soovitused programmi edasiseks arenduseks, et programmi kasutus oleks mugavam või analüüs põhjalikum.

6. Viidatud kirjandus

[1] Leinonen J., Longi K., Klami A., Vihavainen A. Automatic Inference of Programming Performance and Experience from Typing Patterns. *SIGCSE '16: Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 2016. p. 132-137. https://doi.org/10.1145/2839509.2844612

[2] Longi K., Leinonen J., Nygren H., Salmi J., Klami A., Vihavainen A. Identification of programmers from typing patterns. *Koli Calling '15: Proceedings of the 15th Koli Calling Conference on Computing Education Research*. 2015. p. 60-67. https://doi.org/10.1145/2828959.2828960

[3] Lin D.-T. Computer-access authentication with neural network based keystroke identity verification. *Proceedings of International Conference on Neural Networks (ICNN'97)*. 1997. Vol. 1. p. 174-178. https://doi.org/10.1109/ICNN.1997.611659

[4] Cho S., Han C., Han D. H., Kim H.-I. Web-Based Keystroke Dynamics Identity Verification Using Neural Network. *Journal of Organizational Computing and Electronic Commerce*. 2000. Vol 10. Nr 4. p. 295-307. https://doi.org/10.1207/S15327744JOCE1004_07

[5] Edwards J., Leinonen J., Birthare C., Zavgorodniaia A., Hellas A. Programming Versus Natural Language: On the Effect of Context on Typing in CS1. *Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER'20)*. 2020. p. 204-215. https://dl.acm.org/doi/pdf/10.1145/3372782.3406272

[6] Byun J., Park J., Oh A. Detecting Contract Cheaters in Online Programming Classes with Keystroke Dynamics. *Proceedings of the Seventh ACM Conference on Learning @ scale*. 2020. p. 273-276. https://dl.acm.org/doi/abs/10.1145/3386527.3406726

[7] Leping V., Lepp M., Niitsoo M., Tõnisson E., Vene V., Villems A. Python prevails. *CompSysTech 2009: Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. 2009. Nr 87. p. 1-5. https://dl.acm.org/doi/10.1145/1731740.1731833

[8] Annamaa A. Introducing Thonny, a Python IDE for Learning Programming. *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. 2015. p. 117-121. https://doi.org/10.1145/2729094.2754849

[9] Roosi A. Thonny logifailide analüüsi automatiseerimine. TÜ arvutiteaduse instituudi bakalaureusetöö. 2019. http://hdl.handle.net/10062/66233

[10] Python Software Foundation. Python Package Index. https://pypi.org/ (09.04.21)

[11] Mckinney W. pandas: a Foundational Python Library for Data Analysis and Statistics. *Proceedings of the 9th Python in Science Conference*. 2010. Vol. 445. p. 56-61. https://conference.scipy.org/proceedings/scipy2010/mckinney.html

[12] Hunter J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*. 2007. Vol 9. Nr 3. p. 90-95. https://doi.org/10.1109/MCSE.2007.55

[13] Farrell D. DataExplore: An Application for General Data Analysis in Research and Education. *Journal of Open Research Software.* 2016. Vol. 4. Nr 1. p. e9. http://doi.org/10.5334/jors.94

[14] Microsoft. File formats that are supported in Excel. https://support.microsoft.com/en-us/office/file-formats-that-are-supported-in-excel-0943ff2c-6014-4e8d-aaea-b83d51d46247 (11.04.21)

Lisad

I. N-graafide analüsaatori lähtekood

Selle töö raames loodud programmi lähtekood on saadaval Githubis aadressil https://github.com/NuclearPanda/thonnyNGraphs.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Johan Erik Pukk,

1. Annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose

N-graafide analüsaatori loomine,

mille juhendajad on Heidi Meier ja Marina Lepp,

reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Johan Erik Pukk

07.05.2021