

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1487

**Izrada aplikacije za analitiku i predviđanje
rezultata učenja temeljene na intelligentnim
algoritmima**

Roko Krstulović

Zagreb, srpanj 2017.

1. Sadržaj

| | |
|---|----|
| 1. Uvod | 1 |
| 2. Korištene tehnologije | 2 |
| 2.1. Microsoft SQL Server..... | 2 |
| 2.1.1. SQL sintaksa..... | 2 |
| 2.2. JSON | 3 |
| 2.3. Python..... | 4 |
| 3. Author | 6 |
| 3.1. Kolaborativne lekcije | 6 |
| 3.1.1. Oblik zabilješke kolaborativne lekcije..... | 7 |
| 3.2. Kompetitivne lekcije | 8 |
| 3.2.1. Oblik zabilješke kompetitivne lekcije..... | 8 |
| 3.3. Lekcije proširene stvarnosti..... | 9 |
| 3.3.1. Oblik zabilješke lekcije proširene stvarnosti..... | 9 |
| 4. Baza podataka..... | 11 |
| 4.1. Tablica LogEvent | 12 |
| 4.2. Tablica ContextualInfo..... | 13 |
| 4.3. Tablica User..... | 13 |
| 5. Korišteni algoritmi | 14 |
| 5.1. Iterativna metoda konvergencije | 14 |
| 5.2. Nelinearna regresija..... | 19 |
| 6. Arhitektura aplikacije..... | 21 |
| 6.1. Pregled aplikacije..... | 21 |

| | | |
|--------|---|----|
| 6.2. | Skripte za preuzimanje zabilješki | 22 |
| 6.2.1. | Preuzimanje zabilješki kolaborativnih lekcija | 22 |
| 6.2.2. | Preuzimanje zabilješki kompetitivnih lekcija..... | 23 |
| 6.2.3. | Preuzimanje zabilješki lekcija proširene stvarnosti . | 24 |
| 6.3. | Pred-obrada zabilješki..... | 25 |
| 6.3.1. | Pred-obrada zabilješki kolaborativnih lekcija | 26 |
| 6.3.2. | Pred-obrada zabilješki kompetitivnih lekcija..... | 27 |
| 6.3.3. | Pred-obrada zabilješki AR lekcija | 27 |
| 6.4. | Treniranje sustava nad podacima | 28 |
| 6.5. | Evaluacija podataka | 28 |
| 6.6. | Predikcija podataka..... | 29 |
| 6.7. | Vizualizacija rezultata..... | 29 |
| 7. | Upute za korištenje aplikacije..... | 30 |
| 7.1. | Upotreba aplikacije..... | 30 |
| 7.1.1. | Skripta controller.py..... | 30 |
| 7.1.2. | Skripte za predviđanje | 32 |
| 7.1.3. | Skripta display_profile.py | 34 |
| 7.1.4. | Skripta download.py | 35 |
| 7.1.5. | Skripte za uklanjanje posebnih znakova | 35 |
| 7.1.6. | Skripte za uklanjanje navodnika | 36 |
| 7.1.7. | Skripta commas.py | 36 |
| 7.1.8. | Skripta filter.py..... | 36 |
| 7.1.9. | Skripte analyseUsers.py i analyseLessons.py | 36 |

| | | |
|------|---|----|
| 8. | Prikaz rezultata | 39 |
| 8.1. | Prikaz promjene težina zadatka kroz iterativnu metodu konvergencije | 40 |
| 8.2. | Prikaz profila učenika | 41 |
| 8.3. | Prikaz predikcije dobrote učenika..... | 42 |
| 9. | Zaključak | 43 |
| 10. | Sažetak | 44 |
| 11. | Summary | 45 |

Zahvale...

1. Uvod

Ubrzanim razvojem tehnologije svaki se aspekt ljudskog života digitalizira, a tako i obrazovanje. Učenicima nikad nisu bili dostupniji materijali za učenje i vježbanje naučenog nego danas. Učiti mogu bilo gdje, bilo kad i na bilo kojem računalu ili mobilnom uređaju. Kao primjer mogu se uzeti razne android aplikacije namijenjene za učenje poput *Duolingo*, *SoloLearn* itd.

Kako bi se razbila monotonija klasičnih predavanja, za vrijeme nastave učenicima se počinju davati digitalne lekcije. Jedna web aplikacija koja omogućuje kreiranje takvih lekcija je *Author*. Digitalne lekcije *Author-a* već se 2. godinu koriste u jednoj osnovnoj školi. Za vrijeme rješavanja zadataka digitalnih lekcija generiraju se i pohranjuju zabilješke (engl. „log“) akcija učenika. Generirane zabilješke prate napredak učenika te mogu dati učiteljima i roditeljima bolji uvid u napredak njihovog djeteta.

Cilj ovog diplomskog rada je razviti aplikaciju koja će pomoći analitike i pametnih algoritama nad zabilješkama dati bolji uvid u učenje učenika, pružiti vizualizaciju profila učenika generiranih iz zabilješki, te vizualizirati povijest napretka kao i predvidjeti budući napredak.

2. Korištene tehnologije

U ovom će poglavlju biti navedene tehnologije korištene u ovom diplomskom radu.

2.1. Microsoft SQL Server

Micosoft SQL Server je relacijska baza podataka koju je razvio Microsoft. Primarni jezik Microsoft SQL Servera je Transact SQL (T-SQL). Osim klasičnih SQL upita T-SQL pruža i naprednu funkcionalnost poput grananja i neizrazitog (*engl. „fuzzy“*) pretraživanja teksta.

2.1.1. SQL sintaksa

U radu će biti korišteni jednostavni SQL upiti prema bazi, stoga će u ovom poglavlju, u kratim crtama biti objašnjena sintaksa SQL upita koji će biti korišten. Slika 1 prikazuje jednostavan SQL upit preko kojeg će biti objašnjena njegova sintaksa. Upitom se dohvaćaju atributi atr1 i atr2 iz tablice tab1 i atr3 iz tablice tab2 (pogledaj SELECT dio) u kojem je atribut atr3 iz tablice tab1 jednak atributu atr1 iz tablice tab2 (pogledaj JOIN dio) i gdje atributi atr2 iz tablice tab1 i atr3 iz tablice tab2 ispunjavaju uvjet definiran funkcijom uvjet (pogledaj WHERE dio).

```
SELECT
    tab1.atr1,
    tab1.atr2,
    tab2.atr3
FROM
    tab1
JOIN
    tab2 ON tab1.atr3 = tab2.atr1
WHERE
    uvjet(tab1.atr2, tab2.atr3)
```

Slika 1. Primjer SQL upita

U SELECT dijelu upita upisuju se imena stupca tablica (koji se još nazivaju atributi) koje želimo dohvatiti preko forme [ime tablice].[ime stupca] (ime

tablice nije potrebno navoditi ime stupca jedinstveno određuje tablicu).

U FROM dijelu upita upisuje se ime tablice iz koje se dohvaćaju atributi. U slučaju dohvaćanja atributa iz više tablica, sve preostale tablice se zapisuju u zasebne JOIN dijelove u kojima se upisuje ime tablice i po kojem se kriteriju one spajaju s prethodno dodanim tablicama unutar ON dijela.

U WHERE dijelu stavljaju se dodatni uvjeti koje n-torke koje dohvaćamo moraju zadovoljavati. SQL sintaksa omogućava korištenje logičkih operatora AND i OR kojima se mogu pisati složeniji uvjeti u WHERE ili ON dijelu.

2.2. JSON

JSON (od JavaScript Object Notation) je tekstualni zapis programskih objekata. Zabilješke koje sustav obrađuje zapisani su u JSON formatu, stoga će u ovom poglavlju sintaksa JSON-a biti objašnjen u kratkim crtama. Primjer JSON zapisu objekta prikazan je na Slika 2. Objekt omeđuju vitičaste zagrade (linija 1 i linija 12) i sve što se nalazi između njih predstavlja tijelo objekta. Tijelo objekta sastoji se od niza parova ključ-vrijednost odvojene dvotočkom.

```
1  {
2   "atribut1": vrijednost1,
3   "atribut2": [
4     vrijednost2,
5     vrijednost3,
6     vrijednost4
7   ],
8   "atribut3": {
9     "atribut4": vrijednost5,
10    "atribut5": vrijednost6
11  }
12 }
```

Slika 2. Primjer JSON-a

U primjeru, glavni objekt ima tri ključa: atribut1, atribut2 i atribut3 sa njima pridruženim vrijednostima. Vrijednosti mogu biti osnovnog tipa (poput broja ili znakovnog niza) ili složenog tipa (poput niza vrijednosti ili novog objekta). Atributu atribut2 pridružen je niz vrijednosti (lista) dok je atributu atribut3 pridružen novi objekt. Iz razloga što vrijednost pridružena ključu može biti novi objekt ili niz vrijednosti (koje mogu biti ponovno složeni tip) nerijetko se viđaju JSON-i veće dubine.

2.3. Python

Python je široko korišten programski jezik viske razine opće namjene. Python je stvorio Gido van Rossum 1991. kao interpreterski jezik koji potiče čitljivost koda (umjesto uobičajenog kreiranja bloka s vitičastim zagradama u pythonu se blok kreira jednakim uvlačenjem svih susjednih linija koje čine blok) sa sintaksom koja programerima omogućuje jednostavnije izražavanje od klasičnih programskega jezika C++ i Java. Python koristi implicitno tipiziranje (*engl. „duck typing“*) kojim se ne specificiraju kakve tipove podataka varijable mogu pohraniti. Python je idealan za ovaj tip problema u kojemu je potrebno dohvatiti podatke iz baze podataka, parsirati podatke i izvršiti strojno učenje nad podacima, te vizualno prikazati rezultate zbog jednostavnosti dodavanja i dostupnosti paketa sa potrebnom funkcionalnošću.

Korišteni vanjski python paketi su:

- pymssql
- tensorflow
- matplotlib

Paket pymssql se unutar diplomskog rada koristi za spajanje i slanje upita prema Microsoft SQL Server bazi podataka i dohvaćanje istih.

Paket tensorflow je paket za strojno učenje razvijen od Google koji je u radu korišten za predikciju dobrote učenika iskorištavajući simbolički račun i računanje derivacije kompleksnih funkcija. Paket omogućuje izvršavanje na grafičkoj kartici kao i na procesoru ukoliko je potrebna dodatna paralelizacija učenja.

Paket matplotlib je paket za jednostavnu 2D i 3D vizualizaciju podataka korišten za vizualizaciju profila učenika i prikaz krivulje dobivene predikcijom dobrota učenika.

3. Author

Web aplikacija u kojoj su napravljene digitalne lekcije čije se zabilješke obrađuju unutar aplikacije ovog diplomskog rada naziva se Author. Kroz Author je omogućeno stvaranje digitalnih lekcija u obliku prikaznica (*engl. „slide“*) sastavljenih od zadataka koji se prikazuju na mobilnim uređajima i tabletima. Lekcije koje su do sada napravljene u Authoru mogu se po tipu razvrstati u tri kategorije:

- Kolaborativne lekcije
- Kompetitivne lekcije
- Lekcije proširene stvarnosti

Svaki tip lekcija biti će preciznije objašnjen u nadolazećim potpoglavlјima zajedno s oblikom zabilješki njihovih zadataka. Oblik zabilješki se mijenja (i mijenjati će se) kroz vrijeme. Također, zabilješke lekcija istog tipa nemaju identične oblike, pa će biti prikazani samo izabrani predstavnici zabilješka lekcija određenog tipa.

3.1. Kolaborativne lekcije

Kolaborativne lekcije su lekcije u kojima su učenici svrstani u grupe i kao grupe rješavaju zadatke. Grupe se uglavnom sastoje od tri učenika u kojem svatko ima jednu od sljedećih uloga:

- Autor
- Editor
- Kontrolor (*engl. „checker“*)

Kod matematičkih zadataka u kolaborativnim lekcijama, autor čita tekst zadatka i postavlja formulu zadatka. Postavljen zadatak editor rješava i šalje na provjeru kontroloru. Za svakog se učenika generira zabilješka, te sustav svaku zabilješku promatra i ocjenjuje zasebno. Sustav je

napravljen da učenike ocjenjuje na temelju njihovih odluka, a ne na temelju (možda krivih) odluka njihovih kolega (npr. za editora, pošto ne vidi tekst zadatka, ocjenjuje mu se je li dobro riješio (možda krivo) postavljeni zadatak). Zadaci ovog tipa ispituju učenikovu sposobnost rada u timu.

3.1.1. Oblik zabilješke kolaborativne lekcije

Na Slika 3 prikazan je primjer zabilješke zadatka kolaborativne lekcije.

```
{
    "lesson": "638",
    "widget": "1365",
    "logDetails": {
        "inputParams": {
            "isCollaborative": true,
            "isAdaptive": false,
            "groupMembers": [
                "A: IME PREZIME",
                "E: IME2 PREZIME2",
                "C: IME3 PREZIME3"
            ],
            "naslov": "Matematika za 2. razred (Procitajmo i izracunajmo)",
            "lekcija": "formule_2r_mn",
            "tezina": "3",
            "brojPonavljanja": "6",
            "racunskeOperacije": "*/"
        },
        "logEntries": [
            "Lekcija;Tezina;Grupa;RoleConfig;AuthorElapsed Time,EditorElapsed Time,CheckerElapsed Time;AuthorAnswer,EditorAnswer,CheckerAnswer,ProblemNumber,ProblemFormula;formule_2r_mn;3;A: IME PREZIME,E: IME2 PREZIME2,C: IME3 PREZIME3;ACE;42047;5778;2683;4·5;2;C;1;4·5=20"
        ],
        "result": -1
    }
}
```

Slika 3. Primjer zabilješke kolaborativne lekcije

Sustav uz informaciju o učeniku preko zabilješke jednostavno saznaje koju ulogu je učenik imao u rješavanju zadatka i na temelju vrijednosti iz logEntries ocjenjuje učenika. Valja naglasiti da je ovo jedan od oblika zabilješki kolaborativnih lekcija, tj. da postoje drugačiji, a koriste se samo oni koji pružaju informaciju o učeniku i točnosti rješenja, vremenu rješavanja i identifikatoru lekcije.

3.2. Kompetitivne lekcije

Zadaci kompetitivnih lekcija su zadaci koji se postavljaju cijelom razredu u kojem se učenici međusobno natječu u broju točno riješenih zadataka i brzini rješavanja istih. Uz pomoć natjecateljskog duha, podiže se motiviranost učenika u rješavanju zadataka. Obrada zabilješki nastali rješavanjem ovih lekcija učiteljima mogu dati informaciju o sposobnostima učenika u situacijama u kojima je bitna točnost i brzina.

3.2.1. Oblik zabilješke kompetitivne lekcije

Slika 4 prikazuje zabilješku kompetitivne lekcije. Uz informaciju o učeniku sustav preko zabilješke jednostavno dolazi do informacije o točnosti rješenja i vremenu rješavanja.

```
{
    "lesson": "628",
    "widget": "1359",
    "logDetails": {
        "type": "widget_partial_log",
        "data": {
            "inputParams": {
                "isCollaborative": false,
                "isAdaptive": false,
                "naslov": "Virus virus virus virus!",
                "lekcija": "mnS4",
                "tezina": "2",
                "brojPonavljanja": "0",
                "nacinLogiranja": "lpol",
                "trajanje": "15",
                "prikaziVrijeme": true
            },
            "partial_logEntries": [
                {
                    "rbrZapis": 1,
                    "problem": {
                        "firstPart": "20:2=",
                        "secondPart": "10",
                        "thirdPart": "",
                        "correct": true,
                        "unknown": "secondPart",
                        "startTime": 1490086796349,
                        "problemNumber": 1,
                        "endTime": 1490086800518
                    },
                    "vrijeme": 4169,
                    "virusRemainingTime": 9
                }
            ],
            "options": None
        }
    }
}
```

Slika 4. Zabilješka kompetitivne lekcije

Valja naglasiti da je ovo jedan od oblika zabilješki kompetitivnih lekcija, tj. da postoje drugačiji oblici, a koriste se samo oni koji pružaju informaciju o točnosti rješenja, vremenu rješavanja i identifikatoru lekcije.

3.3. Lekcije proširene stvarnosti

Zadaci lekcija proširene stvarnosti od učenika očekuju rješavanje zadatka na neuobičajen način. Primjerice, u zadatku u kojem se traži pronalazak zbroja dvaju brojeva učenik treba izračunati zbroj, rastaviti ga na desetice i jedinice te mobitelom (ili tabletom) slikati karticu s deseticama i karticu s jedinicama. Ovim tipom lekcija pokušava se inovativnošću dodatno motivirati i ocijeniti učenikovu sposobnost prilagodbe.

3.3.1. Oblik zabilješke lekcije proširene stvarnosti

Slika 5 prikazuje primjer zabilješke lekcije proširene stvarnosti. Sustav preko zabilješke čita više pitanja na koja je učenik odgovarao i sve ponuđene odgovore. Zadatak završava trenutkom kad učenik ponudi točan odgovor (a prije toga je možda više puta pogriješio).

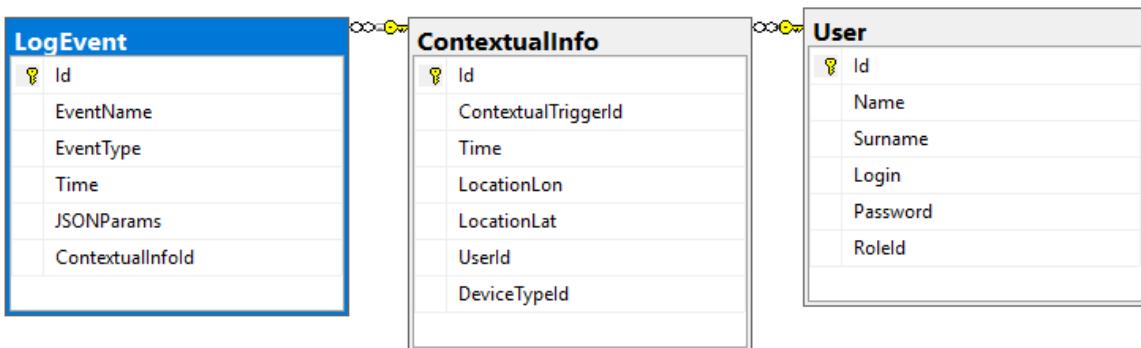
```
[  
  {  
    "task": "Koji poznati pisac je pisao iskljucivo na hrvatskom jeziku?",  
    "correctAnswer": "SENOA",  
    "elapsedSecondsTask": "10378",  
    "answers": [  
      {"elapsedSeconds": "8307", "answer": "5", "correct": "True"}  
    ]  
  },  
  {  
    "task": "Tko od osoba ciji su portreti u razredu je bio vojnik?",  
    "correctAnswer": "JELACIC",  
    "elapsedSecondsTask": "14577",  
    "answers": [  
      {"elapsedSeconds": "0", "answer": "5", "correct": "False"},  
      {"elapsedSeconds": "13482", "answer": "1", "correct": "True"}  
    ]  
  },  
  (...)  
]
```

Slika 5. Primjer dijela zabilješke lekcije proširene stvarnosti

Valja naglasiti da je ovo jedan od oblika zabilješki lekcija proširene stvarnosti, tj. da postoje drugačiji oblici zabilješki, a koriste se samo oni koji pružaju informaciju o točnosti i vremenu rješavanja. Identifikacija lekcije u zabilješkama lekcija proširene stvarnosti razlikuje se od preostalih tipova zabilješki u tome što se kao identifikator zadatka kod zabilješki lekcija proširene stvarnosti koristi pitanje zadatka.

4. Baza podataka

Za analitiku i predviđanje aplikacija koristi podatke spremljene unutar Microsoft SQL Server baze podataka. Zabilješke lekcija pohranjene su unutar tablice LogEvent. Osim zabilješki, aplikacija dohvata informaciju o vremenu zabilješke iz tablice ContextualInfo i podatke o učeniku iz tablice User. Slika 6 prikazuje spomenute tablice i veze među njima preko diagrama.



Slika 6. Diagram tablica LogEvent, ContextualInfo i User

Tablica LogEvent sadrži strani ključ ContextualInfoId prema tablici ContextualInfo kojim se pristupa kontekstualnim informacijama zabilješke, dok tablica ContextualInfo sadrži strani ključ UserId prema tablici User kojim se pristupa podacima učenika čije zabilješke rješavanja zadatka su pohranjene. U sljedećim potpoglavlјima biti će riječ o svakoj od tablica zasebno.

4.1. Tablica *LogEvent*

U tablici LogEvent pohranjuju se podaci o zabilješkama u JSON zapisu koje se stvaraju prilikom rješavanja zadatka s Authora. Tablica se sastoji od sljedećih atributa:

- Id – jedinstvena identifikacija zabilješke
- EventName – ime događaja koji je stvorio zabilješku
- EventType – vrsta događaja koji je stvorio zabilješku
- Time – vrijeme trenutka stvaranja zabilješke na uređaju koji stvara zabilješku
- JSONParams – zabilješka u JSON obliku
- ContextualInfoId – strani ključ na tablicu ContextualInfo

Author podržava više vrsta lekcija, te svaka od lekcija stvara drugačiji oblik zabilješke. Atributi EventName i EventType dobri su ali ne i dovoljni za razlikovanje zabilješki zadatka različitih tipova lekcija, te je potrebno koristiti i atribut JSONParams prilikom filtriranja zabilješki.

Prije obrade potrebno je filtrirati zabilješke jer se u tablici nalazi mnogo zabilješki koje nisu upotrebive za analitiku i predviđanje (poput početka kompetitivne lekcije).

Atribut Time pohranjuje informaciju o stvaranju zabilješke s uređaja koji se koristio prilikom rješavanja zadatka i ono može biti neispravno. Iz tog će se razloga informacija o vremenu izvlačiti iz tablice ContextualInfo.

4.2. Tablica ContextualInfo

U tablici ContextualInfo pohranjeni su podaci o kontekstu zabilješke od kojih će biti korišteni podatak o vremenu iz atributa Time i strani ključ UserId na tablicu User. Atribut Time pohranjuje podatak o ispravnom vremenu zabilješke jer predstavlja vrijeme s uređaja na kojem se nalazi baza podataka. Atribut UserId se koristi za dohvaćanje podataka o učeniku koji je rješavao zadatok od kojeg je dobivena zabilješka.

4.3. Tablica User

U tablici User pohranjeni su podaci o učeniku koji je vlasnik zabilješke. Uz ime i prezime učenika tablica pohranjuje podatke o ulozi, lozinci i posljednjoj prijavi. Aplikacija iz tablice izvlači ime i prezime vlasnika zabilješke koje koristi prilikom obrade zabilješki.

5. Korišteni algoritmi

U ovom će poglavlju biti objašnjeni algoritmi korišteni u ovom diplomskom radu. Iterativna metoda konvergencije je korištena kao numerička metoda rješavanja rekurzivnog problema ovisnosti dobrote učenika i težine zadataka preko tablice riješenosti (tablica koja govori je li učenik A riješio zadatak A) i tablice efikasnosti (tablica koja govori s kojom efikasnosti je učenik A riješio zadatak A). Nelinearna regresija je algoritam strojnog učenja kojim se pronalazi krivulja iz određene familije krivulja koja najbolje aproksimira prethodno izračunate dobrote učenika i koja se koristi za predikciju dobrota učenika u budućnosti.

5.1. Iterativna metoda konvergencije

Računanje dobrote učenika je izrazito težak problem kad nije poznata informacija o težini zadataka (lekcija) koje su rješavali. Slučaj dodatno otežava činjenica da su učenici međusobno rješavali različite zadatke. Uporabom iterativne metode konvergencije izračunate su težine zadataka i dobrote učenika.

Već je rečeno da dobrote učenika ovise o težinama zadataka koje su rješavali. Problem je što i za određivanje težine zadataka potrebno je znati dobrote učenika koji su ih rješavali. Drugim riječima, prisutna je ciklička (rekurzivna) ovisnost. Ideja iterativne metode konvergencije, koja rješava ovu rekurzivnu ovisnost, je prvo učvrstiti dobrote učenika te izračunati težine zadataka, a u sljedećem koraku učvrstiti težine zadataka i izračunati dobrotu učenika i to ponavljati sve dok težine zadataka i dobrote učenika ne izkonvergiraju (dobrote učenika i težine zadataka mogu se gledati kao vektorskim redovima koji konvergiraju prema nekim vrijednostima). Matematička definicija problema izgleda ovako:

$$\begin{aligned}
\mathbf{S}(i+1) &= f(\mathbf{L}(i)), i \geq 0 \\
\mathbf{L}(i+1) &= f(\mathbf{S}(i)), i \geq 0 \\
\lim_{n \rightarrow \infty} \mathbf{S}(n) &= ? \\
\lim_{n \rightarrow \infty} \mathbf{L}(n) &= ?
\end{aligned} \tag{1}$$

gdje su:

\mathbf{S} – vektorski red dobrote učenika

\mathbf{L} – vektorski red težine zadatka

Dobrota j -tog učenika nalazi se unutar vektora $\mathbf{S}(i)$ na poziciji j , gdje je i redni broj koraka metode. Slično, težina zadatka j nalazi se unutar vektora $\mathbf{L}(i)$ na poziciji j , gdje je i redni broj koraka metode.

Dobrota učenika definirana je s realnim brojem iz intervala $[0, 1]$, gdje 1 predstavlja 100-postotnu uspješnost, 0.5 predstavlja 50-postotnu uspješnost (u danjem tekstu učenik prosječne dobrote), dok 0 predstavlja 0-postonu uspješnost nad riješenim zadacima prosječne težine. Težina zadatka je također definirana s realnim brojem iz intervala $[0, 1]$, gdje 1 predstavlja 100-postonu riješenost, 0.5 predstavlja 50-postonu riješenost (u danjem tekstu zadatak prosječne težine), a 0 predstavlja 0-postonu riješenost zadatka od strane prosječnih učenika. Matrica \mathbf{T} na poziciji (i, j) sadrži informaciju o tome je li učenik s indeksom i riješio zadatak s indeksom j na način:

$$\mathbf{T}_{i,j} = \mathbf{T}(i,j) = \begin{cases} -1, & \text{učenik } i \text{ nije uspješno riješio zadatak } j \\ 0, & \text{učenik } i \text{ nije riješavao zadatak } j \\ 1, & \text{učenik } i \text{ je uspješno riješio zadatak } j \end{cases}$$

Preciznost i brzina rješavanja zadaka j od strane učenika i pohranjena je unutar matrice \mathbf{P} na poziciji (i, j) preko realnog broja iz intervala $[0, 1]$, gdje 1 predstavlja najbolju preciznost i najkraće vrijeme. $\mathbf{P}(i, j)$ definirana je kao funkcija preciznosti i brzine rješenja, te je različita za svaki zadatak:

$$\mathbf{P}_{i,j} = \mathbf{P}(i, j) = 1 - \frac{\min(t_{max}, \max(0, t - t_{min} + f * (n - 1)))}{t_{max}} \quad (2)$$

gdje su t , t_{min} , t_{max} , f i n definirani kao:

t – vrijeme koje je bilo potrebno učeniku da riješi zadatak,

t_{min} – očekivano vrijeme za rješavanje zadataka,

t_{max} – maksimalno očekivano vrijeme za rješavanje zadataka,

f – faktor kažnjavanja pogađanja,

n – broj pokušaja

Funkcija ima spomenutu formulu zbog sljedećih svojstava:

- $\mathbf{P}(i, j) = 1$ kad je učeniku trebalo manje od t_{min} vremena da riješi zadatak, a riješio je zadatak iz prvog pokušaja.
- $\mathbf{P}(i, j) = 0$ kad je učeniku trebalo više od t_{max} vremena da riješi zadatak.
- $\mathbf{P}(i^{(k)}, j^{(k)}) > \mathbf{P}(i^{(l)}, j^{(l)})$ kad je učenik k riješio zadatak u manji broj pokušaja od učenika l uz jednako utrošeno vrijeme.

Hiperparametri t_{min} , t_{max} i f imaju drugačije vrijednosti s obzirom na zadatak te ih se treba pametno izabrati (npr. t_{min} se može postaviti kao vrijeme unutar kojeg je 60% učenika riješilo problem, a t_{max} kao dvostruka vrijednost t_{min} , dok f može biti postavljen na 2 sekunde).

Koristeći definirane matrice, prethodno navedeni problem (1) rekurzivne ovisnosti se može izraziti kao:

$$\mathbf{S}(k+1)_i = \frac{\sum_{j=1}^m (f_a(\mathbf{T}_{i,j}) f_t(\mathbf{T}_{i,j}) f_p(\mathbf{L}(k)_j)^{\mathbf{T}_{i,j}} * \mathbf{P}_{i,j})}{\sum_{j=1}^m (f_a(\mathbf{T}_{i,j}) f_t(\mathbf{T}_{i,j}) f_p(\mathbf{L}(k)_j)^{\mathbf{T}_{i,j}} * \mathbf{P}_{i,j})} \quad (3)$$

$$\mathbf{L}(k+1)_j = \frac{\sum_{i=1}^n (f_a(\mathbf{T}_{i,j}) f_t(\mathbf{T}_{i,j}) f_p(\mathbf{S}(k)_i)^{\mathbf{T}_{i,j}} * \mathbf{P}_{i,j})}{\sum_{i=1}^n (f_a(\mathbf{T}_{i,j}) f_t(\mathbf{T}_{i,j}) f_p(\mathbf{S}(k)_i)^{\mathbf{T}_{i,j}} * \mathbf{P}_{i,j})} \quad (4)$$

$$\lim_{k \rightarrow \infty} \mathbf{S}(k) = ?$$

$$\lim_{k \rightarrow \infty} \mathbf{L}(k) = ?$$

gdje je:

m broj zadataka (veličina vektora $\mathbf{L}(i)$),

n broj zadataka (veličina vektora $\mathbf{S}(i)$),

Funkcije f_a , f_t i f_p su pomoćne funkcije sa svrhom izvlačenja informacija o pokušaju, točnosti i brzini rješavanja zadataka. Konkretno, funkcija f_a je definirana s:

$$f_a(x) = x^2 \quad (5)$$

i vraća 1 ako je učenik pokušao riješiti zadatak, a 0 inače. Funkcija f_t je definirana s:

$$f_t(x) = \frac{x+1}{2} \quad (6)$$

i vraća 1 ako je učenik točno riješio zadatak, a 0 inače. Funkcija f_p boduje riješenost zadatka te je definirana s:

$$f_p(x) = k^{1-2x} \quad (7)$$

gdje je k bilo koja konstanta veća ili jednaka od jedan. Funkcija f_p ima sljedeća svojstva (i zbog njih je tako definirana):

- $f_p(0)^{T_{i,j}} = k$, što znači da učenik dobiva k bodova za uspješno riješen zadatak s najvećom težinom (zadatak s težinom 0 znači da je 0% učenika do sad riješilo taj zadatak), a gubi $\frac{1}{k}$ u slučaju da ga je pogriješio.
- $f_p(0.5)^{T_{i,j}} = 1$, što znači da učenik dobiva 1 bod za uspješno riješen prosječan zadatak (zadatak s težinom 0.5 smatra se prosječnim zadatkom), no isto toliko i gubi kad neispravno riješi prosječan zadatak.
- $f_p(1)^{T_{i,j}} = \frac{1}{k}$, što znači da učenik dobiva $\frac{1}{k}$ bod za točno riješen zadatak s najlakšom težinom (zadatak s težinom 1 znači da je 100% učenika do sad riješilo taj zadatak), no gubi k bodova u slučaju da ga pogrešno riješi.

Pomoću funkcije f_p omogućeno je nagrađivanje učenika za rješavanje težih zadataka, te je ublaženo kažnjavanje za greške nad istim. Ovo svojstvo je posebno bitno za situacije kad svi učenici ne rješavaju isti skup zadataka.

Pokazuje se da funkcije f_a , f_t i f_p na jednak način opisuju ovisnost težine zadatka o dobroti učenika koji su ih rješavali. Da bi se mogla iskoristiti iterativna metoda konvergencije, uz rekurzivnu relaciju potrebno je imati i početne vrijednosti. Ovdje se valja zapitati, kad će metoda konvergirati i hoće li uvijek konvergirati u isti par vrijednost vektora S i L ? Konvergencija redova dokazana je iscrpnom pretragom

početnih vrijednosti $S(0)$ i $L(0)$ te matricom T za manji broj učenika i zadataka. Također, za veće bojeve učenika i zadataka pokrenuto je preko milijun simulacija i svaki put je metoda jako brzo (unutar 20 iteracija) izkonvergirala do zadane preciznosti čime je eksperimentalno dokazana tvrdnja da će metoda uvijek izkonvergirati u par vektora S i L . Ipak, eksperimentalno je pokazano da par vektora u koje će izkonvergirati metoda ovisi o odabiru početnih vrijednosti vektora (to je i za očekivati jer za npr. kad svi učenici riješe sve zadatke, dva se jednakobrojno dobri zaključka mogu donijeti: učenici su savladali gradivo ili zadaci su jednostavniji). Kako bi se riješio ne determinizam ovog problema savjetuje se da se za početne vrijednosti dobrote učenika postavi vrijednost 0.8, a težine zadataka 0.5. Tom se pretpostavkom rješenje gura prema tome da sustav kaže da su učenici savladali gradivo ispred zaključka da su zadaci jednostavniji.

5.2. Nelinearna regresija

Regresija je statistički proces u strojnom učenju kojim se određuju parametri krivulje tako da krivulja što bolje aproksimira točke u prostoru. Konkretno, za svaku se točku računa pogreška krivulje za određene parametre, te se odabiru oni parametri koji minimiziraju pogrešku. Za pogrešku se najčešće uzima suma kvadratnih gubitaka:

$$E(h|T) = \frac{1}{2} \sum_{i=1}^N \left(T_y^{(i)} - h(T_x^{(i)}) \right)^2 \quad (8)$$

gdje h predstavlja parametre krivulje (i krivulju s tim parametrima), T predstavlja skup točaka koje krivulja pokušava aproksimirati. Rješenje regresije je skup parametara za koje je ova pogreška najmanja:

$$h^* = \operatorname{argmin}_{h \in H} E(h|\mathbf{T}) = \operatorname{argmin}_{h \in H} \frac{1}{2} \sum_{i=1}^N \left(\mathbf{T}_y^{(i)} - h(\mathbf{T}_x^{(i)}) \right)^2 \quad (9)$$

Zbog jednostavnosti problema, optimizacijski postupak koji je korišten za pronađenje traženih parametara je gradijentni spust. Gradijentni spust koristi informaciju o gradijentu (koji pokazuje smjer najvećeg rasta funkcije) kako bi se iterativno približavao minimumu:

$$\begin{aligned} h^{n+1} &= h^n - \gamma \nabla E(h^n | \mathbf{T}) \\ h^* &= \lim_{n \rightarrow \infty} h^n \end{aligned} \quad (10)$$

gdje je γ stopa učenja. Za dovoljno male stope učenja uz pomoć definicije derivacije može se pokazati da vrijedi:

$$E(h^n | \mathbf{T}) \geq E(h^{n+1} | \mathbf{T}), \forall n \in \mathbb{N} \quad (11)$$

čime je vidljivo da će algoritam uz dovoljno iteracija doći do minimuma. Osim stope učenja u radu je korišten broj iteracija kao hipiperparametar algoritma koji definira nakon koliko iteracija algoritma će algoritam stati.

Familiju krivulja koje su promatrane su funkcije oblika (Stanford-B model krivulje učenja):

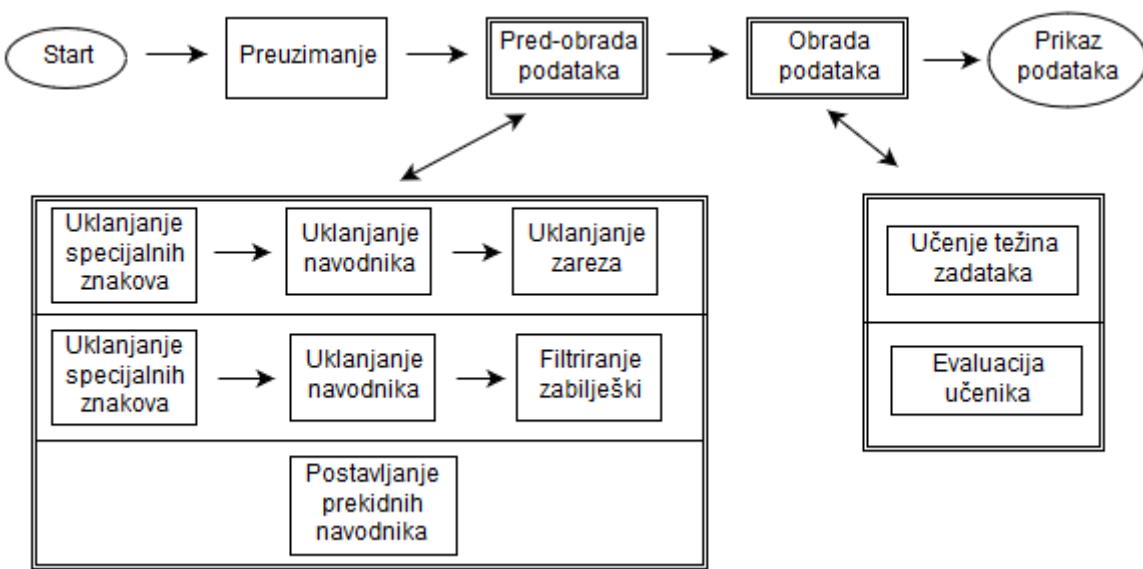
$$f(x) = 1 - C * (x + B)^b \quad (12)$$

gdje parametar B predstavlja iskustvo, b predstavlja stopu učenja, a C predstavlja vrijeme (cijenu) rješavanja prvog zadatka bez prethodnog iskustva. Navedena tri parametra karakteriziraju svakog učenika zasebno, te ih algoritam uči iz njihovih zabilješki. Na temelju naučenih parametra, sustav predviđa budući uspjeh učenika.

6. Arhitektura aplikacije

6.1. Pregled aplikacije

Glavna namjena aplikacije je prikaz profila učenika na temelju zabilješki koji su se skupili njegovim korištenjem sustava. Cjevovod kroz koji teku podaci iz baze do vizualnog prikaza korisniku prikazan je na Slika 7.



Slika 7. Apstraktni prikaz rada aplikacije

Aplikacija prvo preuzima zabilješke konkretnog tipa lekcija sa Microsoft SQL Server baze podataka pomoću SQL upita, koji je karakterističan za tip lekcije, koji se potom pohranjuju i prosljeđuju skripti za pred-obradu podataka. Pred-obrada podataka se razlikuje ovisno o tipu lekcije, te ima ulogu pripreme podataka za obradu. Pripremljeni podaci pohranjuju se i prosljeđuju skripti za obradu podataka. Skripta za obradu podataka izvršava dva različita zadatka: učenje težina zadatka i evaluacija učenika na temelju naučenih težina podataka. Obradom podataka rezultati se pohranjuju i prikazuju korisniku. Dodatno, nakon evaluacije učenika, podaci se mogu proslijediti na predikciju budućeg uspjeha.

Zabilješke različitih tipova lekcija se razlikuju, stoga je za očekivati da će se i svaka od faza unutar cjevovoda razlikovati. Sa slike je vidljivo da predobrada podataka za kolaborativne lekcije nije ista kao i za kompetitivne, odnosno lekcije proširene stvarnosti. Nadalje, s arhitekturom cjevovoda, sustav se može ubrzati koristeći oblikovni obrazac proizvođač-potrošač (engl. „*Producer-customer pattern*“) preko paralelizacije.

6.2. Skripte za preuzimanje zabilješki

Zabilješke je potrebno preuzeti iz baze podataka prije nego što ih se može početi obrađivati. Zabilješke su pohranjene na Microsoft SQL Serveru i dohvaćaju se unutar python skripti pomoću python paketa pymssql. Iz razloga što su zabilješke lekcija različite za lekcije različitog tipa upiti prema bazi se razlikuju. Također, skripte omogućavaju postavljanje korisniku interesantnih vremenskih datuma koje modificiraju upite prema bazi podataka.

U nadolazećim potpoglavlјima biti će objašnjeno dohvaćanje zabilješki za svaki tip lekcija.

6.2.1. Preuzimanje zabilješki kolaborativnih lekcija

Zabilješke kolaborativnih lekcija koji se prikupljaju dok učenici rješavaju zadatke pohranjuju se unutar tablice LogEvent na Microsoft SQL Serveru u JSON obliku. Uz tablicu LogEvent potreban je podatak o vremenu koji se uzima iz tablice ContextualInfo, te ime i prezime učenika za koji se spremaju zabilješke iz tablice User. Skripta *download.py* sadrži logiku izgradnje upita, postavljanje upita prema bazi i zapisivanje rezultata. Tipičan upit prema bazi za zabilješke kolaborativnih lekcija prikazan je na Slika 8.

```

SELECT
    [User].Name,
    LogEvent.Id,
    LogEvent.EventName,
    LogEvent.EventType,
    LogEvent.Time,
    CONVERT(NVARCHAR(MAX), LogEvent.JSONparams),
    LogEvent.ContextualInfoId
FROM
    LogEvent
JOIN
    ContextualInfo ON LogEvent.ContextualInfoId = ContextualInfo.Id
JOIN
    [User] ON ContextualInfo.UserId = [User].Id
WHERE
    JSONparams LIKE '{"lesson":%isCollaborative%}' AND
    eventName = 'widget_log' AND
    (
        (ContextualInfo.Time BETWEEN '6/15/2016' and '6/15/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '6/20/2016' and '6/20/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '7/12/2016' and '7/12/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '10/28/2016' and '10/28/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '11/6/2016' and '11/6/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '11/7/2016' and '11/7/2016 23:59:59')
    )
)

```

Slika 8. Primjer upita za zabilješke kolaborativnih lekcija

Sa slike je vidljivo korištenje tablica LogEvent, ContextualInfo i User preko kojih se dobivaju sve potrebne informacije za daljnju analizu zabilježaka. Sa slike je, također, vidljiv filter koji se filtriraju zabilješke kolaborativnih lekcija kao i datumi za koje se traže. U kasnijem poglavlju će biti objašnjene upute korištenja skripte *download.py*.

6.2.2. Preuzimanje zabilješki kompetitivnih lekcija

Zabilješke kompetitivnih lekcija se isto nalaze u tablici LogEvent na Microsoft SQL Serveru. Na jednak se način kao i kod kolaborativnih lekcija preko tablica ContextualInfo i User dolazi do podataka o datumu zapisa i imenu učenika. Primjer upita koji python skripta *download.py* upućuje bazi za zabilješke kompetitivnih lekcija prikazan je na Slika 9.

```

SELECT
    [User].Name,
    LogEvent.Id,
    LogEvent.EventName,
    LogEvent.EventType,
    LogEvent.Time,
    CONVERT(NVARCHAR(MAX), LogEvent.JSONparams),
    LogEvent.ContextualInfoId
FROM
    LogEvent
JOIN
    ContextualInfo ON LogEvent.ContextualInfoId = ContextualInfo.Id
JOIN
    [User] ON ContextualInfo.UserId = [User].Id
WHERE
    JSONparams LIKE '%{%' AND LogEvent.EventType = 'Player'
    eventName = 'widget_log' AND
    (
        .....
        (ContextualInfo.Time BETWEEN '3/25/2016' and '3/25/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '3/26/2016' and '3/26/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '3/29/2016' and '3/29/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '3/30/2016' and '3/30/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '3/31/2016' and '3/31/2016 23:59:59') OR
        (ContextualInfo.Time BETWEEN '4/1/2016' and '4/1/2016 23:59:59')
    )
)

```

Slika 9. Primjer upita za zabilješke kompetitivnih lekcija

Sa slike je vidljivo da se zabilješke kompetitivnih lekcija dohvaćaju na isti način na koji se dohvaćaju zabilješke kolaborativnih lekcija s razlikom u filteru s ciljem filtriranja isključivo zabilješki kompetitivnih lekcija. U kasnijem poglavlju biti će objašnjen način korištenja skripte *download.py*.

6.2.3. Preuzimanje zabilješki lekcija proširene stvarnosti

Zapisi lekcija proširene stvarnosti također nalaze se nalaze u tablici LogEvent na Microsoft SQL Serveru. Na jednak se način kao i kod prethodnih lekcija preko tablica ContextualInfo i User dolazi do podataka o datumu zapisa i imenu učenika. Preko python skripte *download.py* zabilješke se dohvaćaju iz baze i pohranjuju, a potom, ako je potrebno prosljeđuju na daljnju obradu. Primjer upita koji python skripta *download.py* upućuje bazi za zabilješke lekcija proširene stvarnosti prikazan je na Slika 10.

```

SELECT
    [User].Name,
    LogEvent.Id,
    LogEvent.EventName,
    LogEvent.EventType,
    LogEvent.Time,
    CONVERT(NVARCHAR(MAX), LogEvent.JSONparams),
    LogEvent.ContextualInfoId
FROM
    LogEvent
JOIN
    ContextualInfo ON LogEvent.ContextualInfoId = ContextualInfo.Id
JOIN
    [User] ON ContextualInfo.UserId = [User].Id
WHERE
    JSONparams LIKE '%{%' AND
    LogEvent.EventType LIKE 'AR%'
(
    (ContextualInfo.Time BETWEEN '3/1/2017' and '3/1/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '3/2/2017' and '3/2/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '3/3/2017' and '3/3/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '3/16/2017' and '3/16/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '4/10/2017' and '4/10/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '4/11/2017' and '4/11/2017 23:59:59') OR
    (ContextualInfo.Time BETWEEN '4/12/2017' and '4/12/2017 23:59:59')
)

```

Slika 10. Primjer upita za zabilješke lekcija proširene stvarnosti

Sa slike je vidljivo da je upit za zabilješke lekcija proširene stvarnosti jednak kao i kod prethodnih zabilješki s jedinom razlikom u filteru koji filtrira isključivo zabilješke lekcija proširene stvarnosti. U kasnijem poglavlju će biti objašnjen način korištenja skripte *download.py*.

6.3. Pred-obrada zabilješki

Pred-obrada zabilješki važan je korak u analizi podataka. Izgled zabilješki se tokom vremena mijenja. Neke zabilješke imaju pogreške koje je moguće prepoznati i popraviti dok neke zbog njih nisu upotrebive. Uloga pred-obrade je pripremiti zabilješke kako bi se isti algoritam mogao primijeniti za različite izglede zabilješki i to samo na one na koje je primjenjiv (treba filtrirati nepotrebne zabilješke). Na Slika 7 se vidi da je pred-obrada drugačija za različite tipove zabilješki, te će u sljedećim

potpoglavljima biti riječ o pred-obradi zabilješki zadataka svakog od tipa lekcija.

6.3.1. Pred-obrada zabilješki kolaborativnih lekcija

Pred-obrada zabilješki kolaborativnih lekcija svodi se na uklanjanje nepotrebnih kosih crta (*engl. „back slash“*) koje se povremeno dvostruko generiraju prilikom korištenja prekidnih znakova (*engl. „escaped character“*) unutar Authora, a potom, uklanjanje navodnika koji povremeno znaju obuhvatiti JSON objekte prilikom spremanja zabilješki (razlog tome najvjerojatnije je pretvaranje objekta u znakovni niz, a potom spremanje unutar zabilješke) i konačno uklanjanje (i/ili izmjenom) nepotrebnih zareza koje su posljedica izmjene strukture zapisa.

Logika uklanjanja nepotrebnih kosih crta za kolaborativne lekcije nalazi se unutar python skripte *slashes.py*. Ustanovilo se da je prekidni znak kose crte uvijek neprimjeren osim kad se nalazi unutar HTML taga slike. Na temelju toga, svako dvostruko pojavljivanje kose crte se uklanja ako se ne nalazi unutar HTML taga slike.

Logika za uklanjanje nepotrebnih navodnika nalazi se unutar python skripte *quotes.py*. Ustanovilo se da se ova anomalija pojavljuje jedino prilikom spremanja liste objekata čime je omogućeno jednostavno uklanjanje anomalije (jer je u JSON-u jasno definirano definiranje liste).

Konačno, analizom zabilješki utvrđeno je da postoje zabilješke u kojem je separator znak *točka-zarez*, dok u isto vrijeme postoje zabilješke istog tipa s separatorom *zarez*. Skripta *commas.py* prepoznaje te slučajevе te sve separatore *zarez* pretvara u separator *točka-zarez*.

U kasnijem poglavlju biti će objašnjen način korištenja skripti *slashes.py*, *quotes.py* i *commas.py*.

6.3.2. Pred-obrada zabilješki kompetitivnih lekcija

Pred-obrada kompetitivnih lekcija slično je pred-obradi kolaborativnih lekcija zbog sličnosti zabilješki istih. Sastoji se od uklanjanja kosih crta (engl. „back slash“) koje se povremeno dvostruko generiraju prilikom korištenja prekidnih znakova (engl. „escaped character“) unutar Authora, a zatim, uklanjanje navodnika koji prilikom upisivanja u zabilješke znaju obuhvatiti JSON objekte (razlog je najvjerojatnije pretvaranje objekta u znakovni niz, a potom spremanje unutar zabilješke) i konačno filtriranja nepotrebnih zabilješki. Uklanjanje kosih crta identično je kao i kod kolaborativnih lekcija te se logika nalazi unutar skripte *slashes_player.py*. Uklanjanje nepotrebnih navodnika je također identično kao i kod kolaborativnih lekcija, te se logika nalazi unutar skripte *quotes_player.py*. Filtriranje nepotrebnih zabilješki nalazi se unutar skripte *filter_player.py* u kojoj se parsira zabilješka i potom provjerava posjeduje li sve potrebne informacije. Ovaj korak drastično smanjuje vrijeme izvršavanja učenja nad podacima pošto će se, za vrijeme učenja, zabilješkama višestruko pristupati, a velik dio podataka je nepotreban (ili neiskoristiv). Više o korištenju spomenutih skripti biti će riječ u kasnijem poglavlju.

6.3.3. Pred-obrada zabilješki AR lekcija

Pred-obrada zabilješki lekcija proširene stvarnosti svodi sa na postavljenje prekidnih navodnika (engl. „escaped quotes“). Postoje dva tipa lekcija proširene stvarnosti. Tip lekcija proširene stvarnosti koji od učenika očekuje odgovore na tekstualna pitanja stvara zabilješke koje je potrebno preureediti neposredno prije obrade istih. Unutar tih pitanja zna se dogoditi da se navodi ime novina, opera ili neke treće iz povijesti poznate činjenice koje su stavljene pod neprekidne znakove navodnika čime je onemogućeno parsiranje JSON-a. U skripti *quotes_AR.py* nalazi se logika kojom se potrebni navodnici zamjenjuju s prekidnim znakovima

navodnika. Više o korištenju skripte *quotes_AR.py* biti će riječ u kasnijem poglavlju.

6.4. Treniranje sustava nad podacima

Podaci, nakon što su pred-obrađeni, se mogu poslati na treniranje sustava u kojem se uče težine zadatka ili na evaluaciju (ocjenjivanje) dobrote učenika. Treniranje sustava idejno je jednako za sve tipove lekcija. Jedina razlika je u tome što se podaci o uspješnom rješavanju zadatka dohvaćaju na različite načine jer su zabilješke svake od tipova lekcija različite. Skripta za treniranje sustava nad podacima je *Lesson_user.py*. Ona naizmjenice poziva skriptu *analyseLessons.py* i *analyseUsers.py* koje računaju težine lekcija uz ne mijenjanje dobrote učenika, odnosno računaju dobrotu učenika uz fiksne težine lekcija. Time je iskorištena iterativna metoda konvergencije kako bi se riješilo rekurzivne (cikličke) ovisnosti dobrote učenika i dobrote zadatka. Skripte *analyseLessons.py* odnosno *analyseUsers.py* dohvaćaju podatke, parsiraju ih, te na temelju svake zabilješke osvježavaju procjenu težine lekcije odnosno dobrote učenika po formulama (3) i (4), i konačno pohranjuju rezultate. Više o korištenju spomenutih skripta biti će riječ u kasnijem poglavlju.

6.5. Evaluacija podataka

Evaluacija podataka slična je treniranju sustava s razlikom da se procjenjuju dobrote učenika na temelju izračunatih težina zadatka lekcija. Težine lekcija pohranjene su unutar datoteke koja se prosljeđuje python skripti *analyseUsers.py* u kojoj se nalazi logika evaluacije dobrote učenika. Skripta *analyseUsers.py* parsira pred-obrađene zabilješke te uz pomoć formule (3) računa dobrotu učenika na temelju izračunatih težina lekcija. Skripta *analyseUsers.py* kao parametar očekuje i datume za koje će izračunati dobrote učenika (za svaki se datum posebno evaluira dobrota

učenika) koji se mogu koristiti za vizualizaciju i predikciju učenikove dobrote. Više o korištenju skripte *analyseUsers.py* biti će riječ u kasnijem poglavlju.

6.6. Predikcija podataka

Nakon što je evaluacija podataka učenika obavljena sve predispozicije za predikciju dobrote učenika su postignute. Prije same predikcije potrebno je izvući podatke o konkretnom učeniku za definirane datume i pohraniti ih kao međurezultat (jer će se ti podaci upotrebljavati velik broj puta prilikom predikcije). Logika predikcije nalazi se unutar python skripti *preprocess_predict.py* i *predict.py*. Skripta *preprocess_predict.py* pohranjuje podatke o učeniku za definirane datume te ih proslijedi skripti *predict.py* koja korištenjem python paketa tensorflow pronalazi krivulju učenja koja najbolje opisuje dobivene podatke upotrebom nelinearne regresije. Više o korištenju spomenutih skripti biti će riječ u kasnijem poglavlju.

6.7. Vizualizacija rezultata

Skripte *analyseUsers.py* (za sve tipove), *analyseLessons.py* (za sve tipove), *predict.py* i *display_profile.py* korisniku omogućuju vizualizaciju rezultata. Skripte *analyseUsers.py* i *analyseLessons.py* vizualno prikazuju izračunate dobrote učenika i težine zadataka, skripta *predict.py* prikazuje rezultat nelinearne regresije nad zadanim dobrotama učenika, a skripta *display_profile.py* prikazuje profil učenika generiran iz pred-obrađenih podataka. Više u upotrebi navedenih skripti bit će u 7. poglavlju dok se primjeri vizualizacija nalaze u poglavlju 8.

7. Upute za korištenje aplikacije

U ovom će se poglavlju opisati korištenje aplikacije. U potpoglavlju 7.1. će se opisati korištenje kontrolne skripte *controller.py*, skripti koje ona koristi i skripte za predviđanje *preprocess_predict.py* i *predict.py*.

7.1. Upotreba aplikacije

7.1.1. Skripta controller.py

Skripta *controller.py* kontrolirajući i prosljeđujući naredbe preostalim skriptama omogućava jednostavno treniranje sustava i evaluaciju zabilješki svih tipova lekcija. Na Slika 11 je prikazan način korištenja spomenute skripte.

```
C:\ Select C:\WINDOWS\system32\cmd.exe

C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python controller.py -h
usage: controller.py [-h] [-f F] [-ip IP] [-port PORT] [-db DB] (-t T | -e)
                      type starting_date ending_date

Controller script that is used for downloading, filtering and training system
/ evaluating users.

positional arguments:
  type            type of lessons. Allowed types: collaborative, competitive,
                  AR
  starting_date  Starting date (dd.mm.YYYY)
  ending_date    Ending date (dd.mm.YYYY)

optional arguments:
  -h, --help      show this help message and exit
  -f F           Take first F elements
  -ip IP         IP address of the database
  -port PORT     PORT of the database
  -db DB         Database name
  -t T           calculate lessons/users (train system)
  -e             evaluate users' score
```

Slika 11. Upute za korištenje python skripte controller.py

Skripta kao argument type očekuje tip zabilješki lekcije koje će se obrađivati. Dozvoljeni tipovi zabilješki su: kolaborativni (collaborative), kompetitivni (competitive) i proširena stvarnost (AR). Osim tipa, skripti je potrebno poslati i vremenski period unutar kojeg se nalaze zabilješke

preko parametara `starting_date` (početak) i `ending_date` (završetak) u zadanim formatu.

Dodatno, potrebno je odrediti hoće li se trenirati sustav (računati težine zadataka) ili će se evaluirati dobrote učenika. To se postiže predajom jednog od dva međusobno isključiva parametra `-t` i `-e`. Parametar `-t` se koristi za treniranje sustava, te mu je još potrebno zadati broj iteracija koje će se izvršiti prilikom treniranja, dok se parametar `-e` koristi za evaluaciju učenika za zadani tip lekcija u određenom vremenskom periodu.

Osim navedenih obveznih parametara skripti se mogu poslati i podaci IP adrese, port-a i imena Microsoft SQL Server baze podataka preko optionalnih parametara `-ip`, `-port` i `-db`. Ako korisnik ne unese te parametre koristiti će se unaprijed zadani parametri. Nadalje, skripta omogućuje korištenje prvih f datuma iz zadanoog vremenskog perioda upotreboom parametar `-f`.

Pokretanjem skripte `controller.py`, preko skripte `get_dates.py`, sa servera se dohvaćaju karakteristični datumi za tip lekcije, iz zadanoog vremenskog perioda, koji će se koristiti unutar svih skripti koje `controller.py` poziva. Dobiveni datumi se spremaju u datoteku s karakterističnim imenom kako bi se prije sljedećeg upita provjerilo postoje li već rezultati zadanoog upita i iskoristili ukoliko postoje. Nakon dohvaćanja datuma, na temelju parametara o treniranju sustava, odnosno evaluaciji učenika, skripta će pokrenuti skriptu `download.py` kako bi preuzela zabilješke za dobivene datume. Ako je od skripte zatraženo da se trenira sustav, zabilješke će biti spremljene kao cjelina, dok će se kod evaluacije učenika zabilješke spremati odvojeno po datumima. Skripte koje poziva `controller.py` međusobno komuniciraju preko datoteka, te im se kroz parametre specificiraju imena datoteka za komunikaciju. Nakon izvršavanja skripte `download.py` pokreću se skripte za pred-obradu ovisno o tipu i nakon toga

skripte za treniranje sustava (naizmjenice skripte *analyseUsers.py* i *analyseLessons.py*) ili evaluaciju dobrote učenika (skripta *analyseUsers.py*).

7.1.2. Skripte za predviđanje

Skripte za predviđanje se koriste nakon što se izvrši evaluacija dobrote učenika kako bi se prikazala krivulja koja najbolje aproksimira izračunate točke dobrote učenika u ovisnosti o rednom broju rješavanja zadataka lekcija istog tipa. Na taj se način određuju najvjerojatnije vrijednosti dobrote učenika u budućnosti. Uloga skripte *preprocess_predict.py* je skupiti podatke o dobrotama za svakog učenika zasebno, po zadanim datumima. Tako će se skripti *predict.py* svi podaci o učeniku koje koristi za predviđanje skupiti u jednu datoteku. Na Slika 12 su prikazane upute za korištenje skripte *preprocess_predict.py*.

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python preprocess_predict.py -h
usage: preprocess_predict.py [-h] type date [date ...]

script that extracts students data from by date stored data and stores it with
previous extracted data if it exists (not nessessary from same dates)

positional arguments:
  type      type of lessons. Allowed types: collaborative, competitive, AR
  date      dates (dd.mm.YYYY)

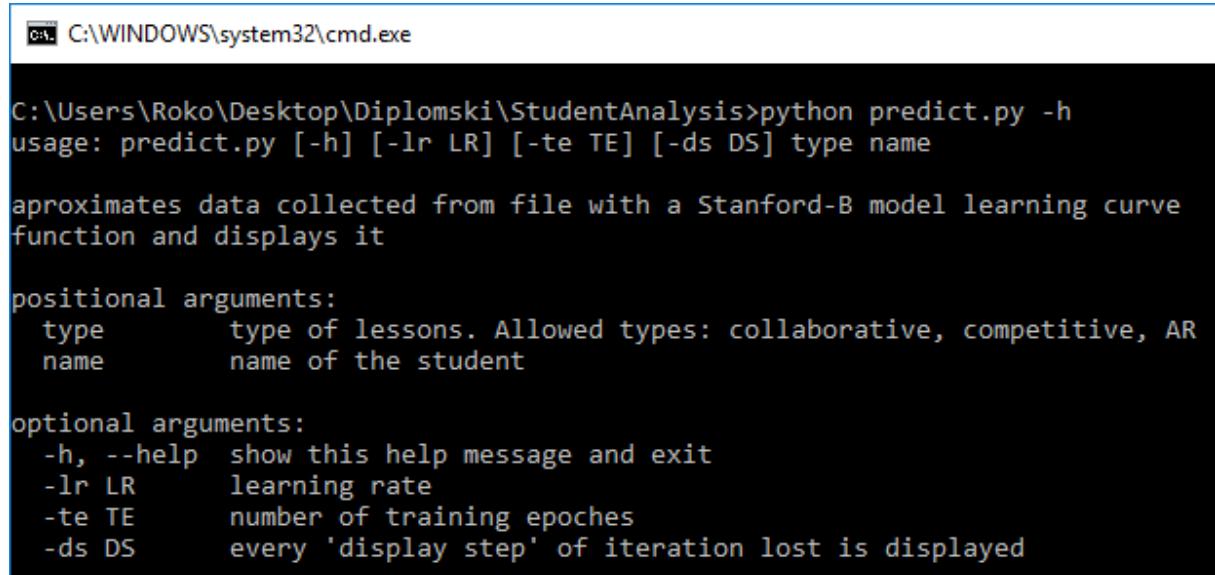
optional arguments:
  -h, --help  show this help message and exit
```

Slika 12. Upute za korištenje skripte *preprocess_predict.py*

Skripta *preprocess_predict.py* preko parametra *type* očekuje tip lekcije čiji se podaci žele grupirati po učenicima. Osim parametra *type* skripta očekuje jedan ili više datuma. Datumi i tip lekcija se koriste za pronalaženje datoteka s podacima o dobrotama učenika za specificirane datume i tip lekcije koji će se upotrijebljavati. Podaci se izvlače i pohranjuju s prethodno izvučenim podacima (ako postoje), bez duplicitiranja podataka

u datoteku s karakterističnim imenom (kako bi skripta *predict.py* znala gdje tražiti podatke). Prethodno prikupljeni podaci nisu nužno iz istih datuma, čime se osigurava da se za predikciju koristi što više podataka.

Skripta *predict.py* koristi podatke dobivene izvršavanjem skripte *preprocess_predict.py*. Upute korištenja skripte *predict.py* prikazane su na Slika 13.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python predict.py -h
usage: predict.py [-h] [-lr LR] [-te TE] [-ds DS] type name

approximates data collected from file with a Stanford-B model learning curve
function and displays it

positional arguments:
  type      type of lessons. Allowed types: collaborative, competitive, AR
  name      name of the student

optional arguments:
  -h, --help  show this help message and exit
  -lr LR     learning rate
  -te TE     number of training epoches
  -ds DS     every 'display step' of iteration lost is displayed
```

Slika 13. Upute korištenja skripte *preprocess_predict.py*

Očekivani parametri skripte *predict.py* su *type* - tip lekcije i *name* - ime učenika čiju dobrotu se želi aproksimirati Stanford-B modelom krivulje učenja (12). Predikcija se obavlja korištenjem python-ovog paketa tensorflow koji obavlja nelinearnu regresiju uz pomoć simboličkog računa.

Parametre algoritma nelinearne regresije (faktor učenja, broj epoha učenja i frekvenciju ispisa) moguće je specificirati kroz argumente *-lr*, *-te* i *-ds*. Rezultati se nakon izvršavanja vizualno prikazuju na grafu s dobivenom krivuljom i točkama dobrote iz koje je krivulja dobivena (više o vizualizaciji u 8. poglavljju).

7.1.3. Skripta *display_profile.py*

Skripta *display_profile.py* se koristi za prikazivanje profila učenika. Prije izvršavanja skripte potrebno je pokrenuti skriptu *preprocess_predict.py* koja skuplja i grupira podatke po učenicima. Iz grupiranih podataka za specificiranog učenika skripta *display_profile.py* izračunava srednju vrijednost svakog tipa i vizualno ih prikazuje preko tkz. radar grafova (engl. „radar charts“).

Slika 14 prikazuje upute za korištenje skripte *display_profile.py*.

```
C:\WINDOWS\system32\cmd.exe
C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python display_profile.py -h
usage: display_profile.py [-h] name [name ...]

calculates and displays student profile

positional arguments:
  name        name of the student

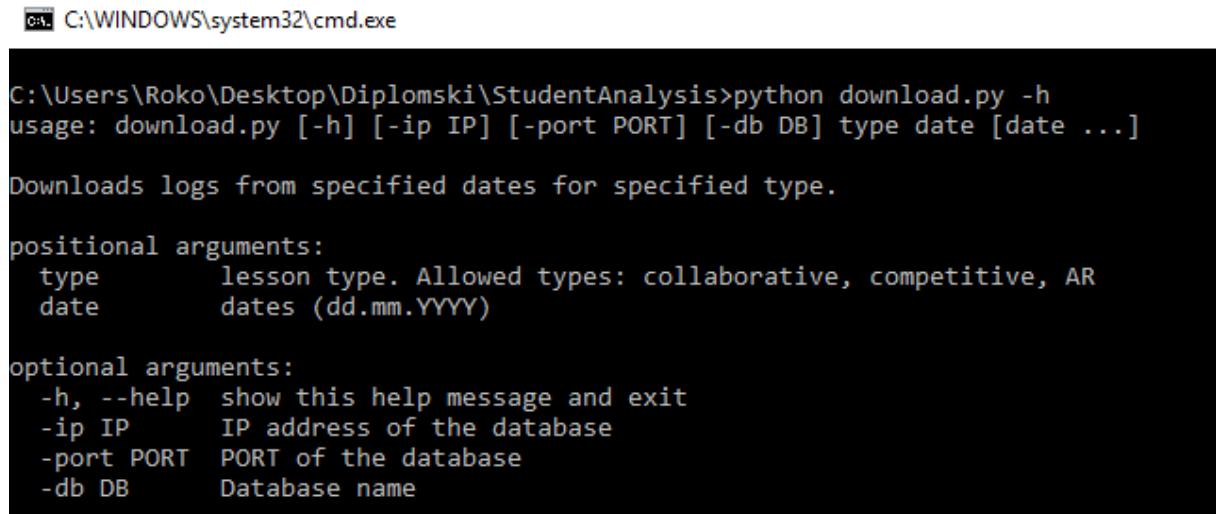
optional arguments:
  -h, --help  show this help message and exit
```

Slika 14. Upute za korištenje skripte *display_profile.py*

Sa slike je vidljivo da skripta očekuje jedno ili više imena učenika čiji se profili žele prikazati. Specificiranjem više imena, profili učenika se prikazuju u istom radar grafu obojani drugačijim bojama što omogućuje jednostavnu usporedbu učenika. Više o vizualizaciji se nalazi u poglavljju 8.

7.1.4. Skripta download.py

Skripta *download.py* se koristi kako bi se preuzele zabilješke zadataka lekcija. Slika 15 prikazuje upute za korištenje skripte *download.py*.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python download.py -h
usage: download.py [-h] [-ip IP] [-port PORT] [-db DB] type date [date ...]

Downloads logs from specified dates for specified type.

positional arguments:
  type      lesson type. Allowed types: collaborative, competitive, AR
  date      dates (dd.mm.YYYY)

optional arguments:
  -h, --help  show this help message and exit
  -ip IP     IP address of the database
  -port PORT PORT of the database
  -db DB     Database name
```

Slika 15. Upute za korištenje skripte *download.py*

Sa slike je vidljivo da skripta *download.py* kao obvezne parametre očekuje tip lekcije preko parametra *type* i jedan ili više datuma preko parametra *date*. Uz neobvezne parametre korisniku je omogućen unos IP adrese, porta i ime baze podataka ako ne želi da skripta koristi unaprijed definiranu bazu. Podaci o korisniku baze i lozinci se čitaju iz datoteke s imenom *config.txt* koja se nalazi u istom direktoriju kao i skripta. Ovisno o tipu i datumima generira se upit prema bazi te se rezultati spremaju u datoteku s imenom koje ovisi o datumima i tipu lekcije.

7.1.5. Skripte za uklanjanje posebnih znakova

Skripte *slashes.py* i *slashes_player.py* služe za pred-obradu zabilješki kojima se uklanjaju nepotrebne kose crte kolaborativnih i kompetitivnih lekcija. Neke od zabilješki unutar baze imaju dvostruko prekidne znakove (engl. „escaped characters“) koje je potrebno ispraviti. Obradom zabilješki pokazalo da se problem može riješiti uklanjanjem svih kosih crta koje se ne nalaze unutar HTML taga slike, što izvršavaju navedene skripte.

Skripte se pozivaju s parametrom jednog ili više datuma, a ime skripte određuje kojeg su tipa zabilješke koje se moraju preureediti.

7.1.6. Skripte za uklanjanje navodnika

Skripte *quotes.py* i *qoutes_player.py* služe za uklanjanje navodnika koje se greškom generiraju prilikom spremanja zabilježaka. Analizom zabilješki utvrđeno je da se problem pojavljuje prilikom prebacivanja liste u JSON format. Skripte se pozivaju parametrom jednog ili više datuma, a ime skripte određuje kojeg su tipa zabilješke koje se trebaju preureediti.

7.1.7. Skripta commas.py

Skripta *commas.py* koristi se za pred-obradu zabilješki kojom se prilagođava dio kolaborativnih zabilješki na način da separator unutar dijela zabilješke promijeni iz zareza u točku-zarez i omogući isti način parsiranja zabilješki u daljnjoj obradi podataka. Skripti se kao parametar šalje jedan ili više datuma preko kojih skripta određuje datoteku koju će preureediti i datoteku u koju će spremiti rezultate.

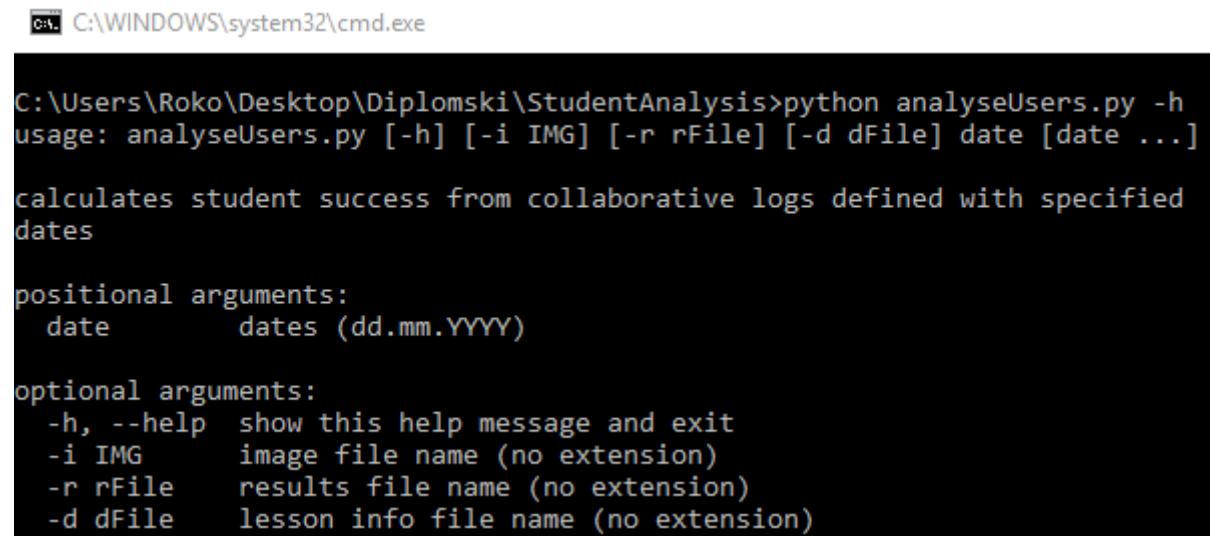
7.1.8. Skripta filter.py

Skripta *filter.py* koristi se za pred-obradu zabilješki kompetitivnih lekcija. Kompetitivni zadaci generiraju mnogo zabilješka od kojih većina nije upotrebljiva za treniranje sustava i evaluaciju dobrote učenika. Ti zabilješci se izbacuju skriptom *filter.py*. Skripti se kao parametar šalje jedan ili više datuma preko kojih skripta određuje datoteku koju treba preureediti i datoteku u koju će se spremiti rezultati.

7.1.9. Skripte analyseUsers.py i analyseLessons.py

Skripte *analyseUsers.py* i *analyseLessons.py* računaju dobrote učenika iz težina lekcija preko formule (3) i težine lekcije iz dobrote učenika preko formule (4). Prolaskom kroz parametrima definirane datoteke skripte čitaju zabilješke, parsiraju ih, izvlače informacije o rješenosti zadataka i njihovom vremenu, a potom ocjenjuju dobrote učenika odnosno težine

lekcija. Upute za korištenje skripte *analyseUsers.py* analogno je uputama za korištenje skripte *analyseLessons.py* i prikazano je na Slika 16.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Roko\Desktop\Diplomski\StudentAnalysis>python analyseUsers.py -h
usage: analyseUsers.py [-h] [-i IMG] [-r rFile] [-d dFile] date [date ...]

calculates student success from collaborative logs defined with specified
dates

positional arguments:
  date      dates (dd.mm.YYYY)

optional arguments:
  -h, --help  show this help message and exit
  -i IMG    image file name (no extension)
  -r rFile  results file name (no extension)
  -d dFile  lesson info file name (no extension)
```

Slika 16. Upute za korištenje skripte *analyseUsers.py*

Skripta kao obvezne parametre očekuje jedan ili više datuma kroz parametar date. Dodatno, skripti se može definirati ime datoteke u koju će vizualno spremati promjene dobrota učenika kroz parametar -i. Imenu datoteke se na kraj dodaje broj iteracije kako ne bi došlo do brisanja međurezultata. Skripti se također preko parametara -r i -d definiraju imena datoteke u koje se spremaju rezultati (dobrote učenika) i datoteke iz koje se čitaju vrijednosti težina zadataka. Parametri skripte *analyseLessons.py* su identični spomenutim parametrima s razlikom da su rezultati težine lekcija, a ulazi dobrote učenika.

Spomenute skripte se koriste za analizu zabilješki kolaborativnih zadataka. Imena skripti za analizu kompetitivnih lekcija i lekcija proširene stvarnosti imaju sličan imena (imena im završavaju s *_player* i *_AR*) i idejno su jednake skriptama za analizu kolaborativnih zabilješki s razlikom u parsiranju zabilješki. Iz razloga što lekcije s proširenom stvarnosti uvijek završavaju s točnim odgovorom (dozvoljen je više pokušaja odgovaranja)

efikasnost učenika se razlikuje od analize kompetitivnih i kolaborativnih lekcija u tome što se unutar formule (2) koristi parametar $n \geq 1$.

8. Prikaz rezultata

Korištenjem lekcija Authora generiraju se i spremaju zabilješke u bazu podataka. Uz prikupljenu dovoljnu količinu podataka, koji daju informacije o učinku i napretku učenika, moguće je izraditi profil učenika i predvidjeti njegov napredak na temelju podataka iz zabilješki. Od korisnika se očekuje logično grupiranje podataka iz zabilješki te uklanjanje nerelevantnih zabilješki (šuma) kako bi se dobili iskoristivi podaci. Primjer logičkog grupiranja podataka je grupiranje zabilješki lekcija istog tipa ili grupiranje zabilješki istih lekcija. Nadalje, prilikom rješavanja zadataka moglo je doći do pada sustava ili se na neki drugi način mogla poremetiti koncentraciji učenika čime su generirani nerelevantni podaci (šum) koje je potrebno ukloniti. Uobičajeni algoritmi za uklanjanje šuma je detekcija primjera koji odskaču (*engl. „outliers“*) preko grupiranja¹ (*engl. „clustering“*) ili korištenjem filtra (npr. Kalmanov² filter).

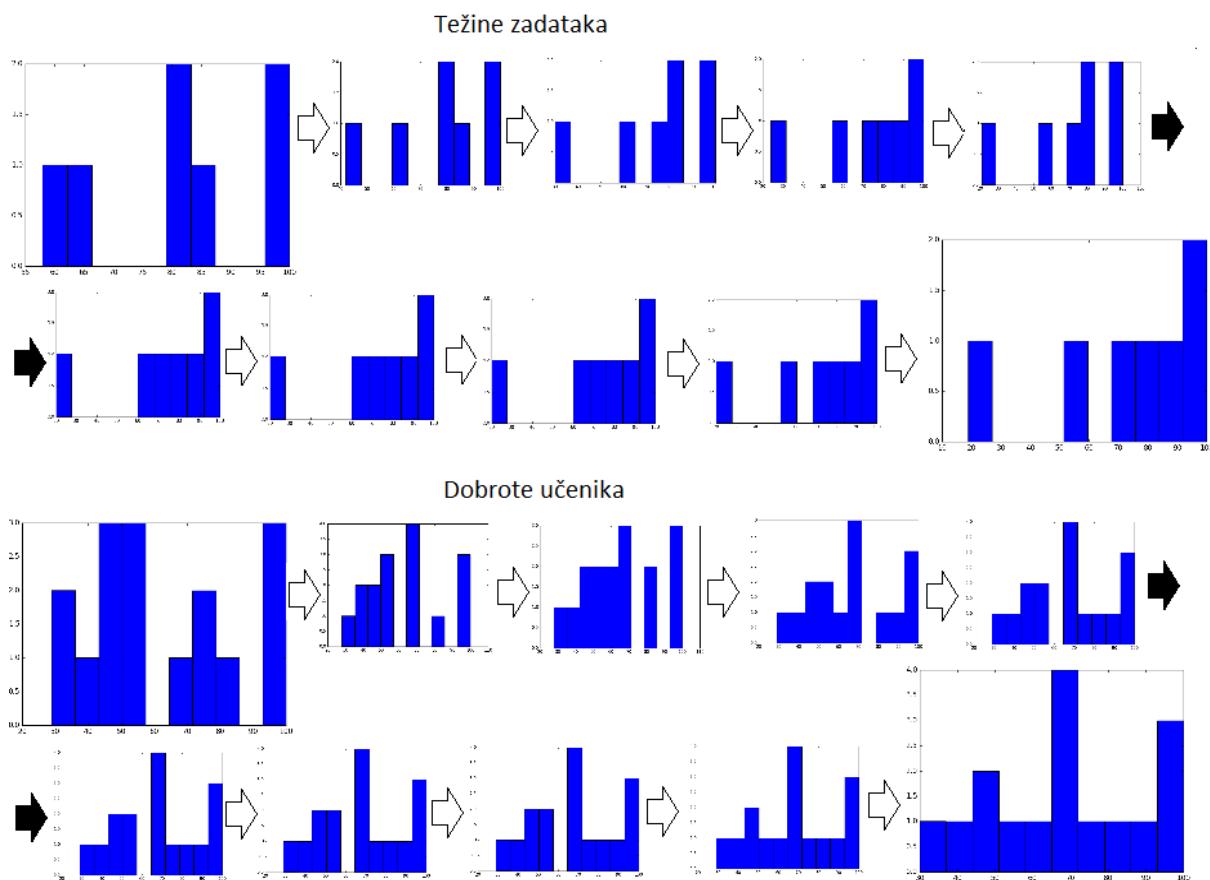
U sljedećim potpoglavlјima biti će prikazana promjena težina zadataka kroz iteracije zajedno s promjenom dobrote učenika, prikaz dobivanja profila učenika i prikaz usporedbe više profila učenika i konačno prikaz predikcije dobrote učenika nakon što su se u aplikaciji za svaki tip lekcija učile težine zadataka nad prvih deset datuma kroz deset iteracija, a potom evaluirali dobrote učenika za svaki datum za koji postoji barem jedna zabilješka.

¹ <http://www.dcc.fc.up.pt/~ltorgo/Papers/ODCM.pdf>

² http://acds-lab.gatech.edu/papers/IROS2007_RobustKF.pdf

8.1. Prikaz promjene težina zadatka kroz iterativnu metodu konvergencije

Računanje težina zadatka lekcija i dobrote učenika koristeći zabilješke, aplikacija rješava uporabom iterativne metode konvergencije. U svakoj se iteraciji težine približavaju težinama koje zadovoljavaju rekurzivni problem definiran jednadžbama (1). Prikaz promjena težina uz promjene dobrote učenika prikazane su na Slika 17.



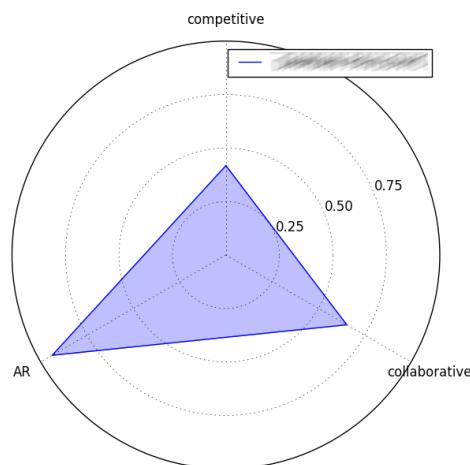
Slika 17. Težina zadatka i dobrote učenika za vrijeme iterativne metode konvergencije (kolaborativni zadaci)

U početnom stanju težine svih zadatka iznose 0.5 (graf bi imao samo jednu kolonu). Iteracijama se odvajaju teži zadaci (lijevi) od lakših zadatka (desni). Zbog toga se svakom iteracijom drugačije ocjenjuju dobrote učenika.

Sa slike je vidljivo da je metoda već nakon 4. iteracije blizu stabilnom stanju (jer su promjene postale zanemarive).

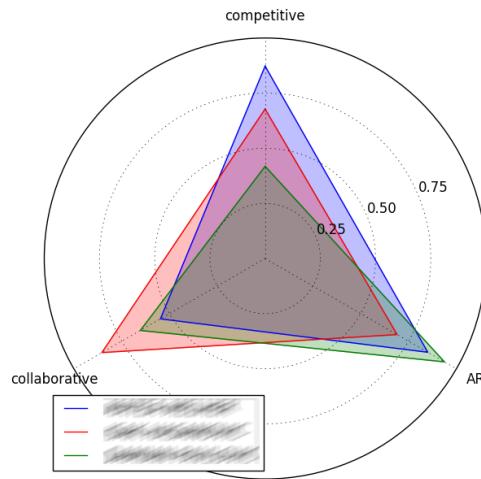
8.2. Prikaz profila učenika

Rješavanjem zadataka učenik generira zabilješke čiji broj brzo naraste. Aplikacija koristeći zabilješke može izračunati i vizualno prikazati profil učenika što pruža puno bolji uvid u stanje učenika. Na Slika 18 prikazan je profil jednog učenika. Rezultati prikazani na slici dobiveni su uprosječivanjem svih rezultata zabilješki lekcija istog tipa.



Slika 18. Profil učenika nakon rješavanja virtualnih zadataka

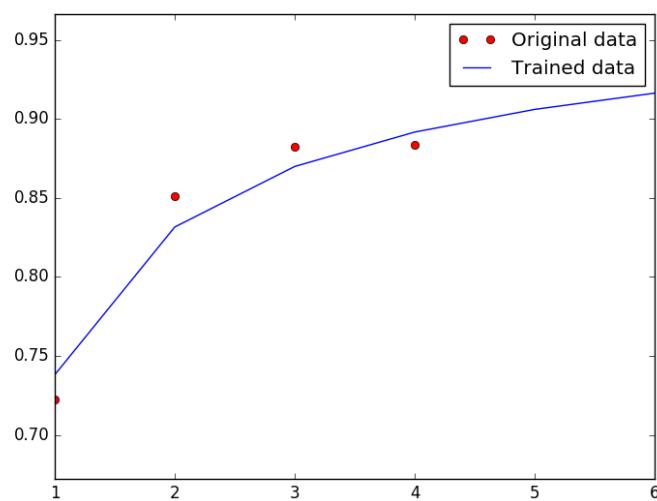
Osim profila jednog učenika, aplikacija omogućuje prikazivanje više učenika na istom grafu čime čini usporedbu učenika jednostavnijom. Slika 19 prikazuje usporedbu prijašnjeg učenika s dva nova učenika.



Slika 19. Usporedba profila triju učenika

8.3. Prikaz predikcije dobrote učenika

Poboljšanje dobrote učenika očekuje se svakim novim rješavanjem zadataka sličnog tipa. Aplikacija omogućuje aproksimiranje izračunatih dobrota učenika iz zadataka istog tipa lekcije Stanford-B modelom krivulje učenja opisane u prijašnjem poglavlju čime se radi predikcija dobrote učenika. Primjer predikcije prikazan je na Slika 20/Slika 19.



Slika 20. Predikcija dobrote učenika na kompetitivnoj lekciji

9. Zaključak

Digitalne lekcije imaju sve veću ulogu u školama i time omogućuju potpuno novi pogled na način podučavanja. Dok učenici vježbaju preko digitalnih lekcija generiraju se zabilješke koje opisuju učenikov pristup i znanje prilikom vježbanja. Dugotrajnim korištenjem digitalnih lekcija gomila se broj zabilješki koje je moguće obraditi algoritmima „velikih količina podataka“ (*engl. „Big Data“*).

Kroz ovaj diplomski rad implementirana je aplikacija koja analizirajući zabilješke rješavanja zadatka učenika, učiteljima (i roditeljima) omogućuje vizualni prikaz profila učenika, usporedbu učenika i uvid u predikciju napretka učenika. Profili mogu biti izrazito korisni učiteljima koji su došli na zamjenu jer bi omogućili brzi uvid u kompetencije učenika.

U radu su korišteni algoritmi nelinearne regresije i metode iterativne konvergencije kojima se izračunala krivulja učenja iz generiranih zabilješki i razriješila ciklička ovisnost dobrote učenika i težina učenika čime je omogućeno ocjenjivanje težina zadatka i ne klasično vrednovanje učenika u kojem težina zadatka ima utjecaj na ocjenu učenika.

10. Sažetak

U ovom diplomskom radu implementirana je aplikacija koja analizom zabilješki generiranih rješavanjem zadataka triju tipova lekcija omogućuje vizualni uvid i usporedbu profila učenika, uz predikciju njihovog napretka rješavanjem zadataka istog tipa. Aplikacija koristeći algoritam iterativne metode konvergencije razrješava cikličku ovisnost dobrote učenika i težine zadataka čime omogućuje ne klasično vrednovanje učenika na temelju težina zadataka.

11. Summary

The main goal of this master thesis was to implement an application which by analysing logs, that were generated while solving problems from three different types of lectures, gives visual insight in student's profile or comparison with other students, along with prediction in student's improvement in solving same type of problems. By using the algorithm of iterative convergence method, application solved recursive dependency between the success of student and weight of the problem, thus allowing non-classical student evaluation considering problem weights.