Dubinska analiza tokova podataka u području edukacije

Blašković, Mihaela

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike – modul Poslovna informatika

Mihaela Blašković

Dubinska analiza tokova podataka u području edukacije

Diplomski rad

Mentor: Prof. dr. sc. Maja Matetić

Rijeka, rujan 2021.

ZADATAK DIPLOMSKOG RADA

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 6.6.2019.

Zadatak za diplomski rad

Pristupnik: Mihaela Blašković

Naziv diplomskog rada: Dubinska analiza tokova podataka u području edukacije

Naziv diplomskog rada na eng. jeziku: Mining of data streams in the field of education

Sadržaj zadatka:

U sustavu za e-učenje generira se tok zapisa aktivnosti studenta pri interakciji sa materijalima za učenje, provjerama znanja, video lekcijama i drugim resursima. Veliki volumen nastalih sekvencijalnih podataka možemo koristiti za modeliranje ponašanja studenta. Zadatak diplomskog rada je primjenom neuronske mreže s povratnom vezom (engl. Recurrent Neural Network) modelirati ponašanje studenta i predvidjeti njegovu aktivnost i uspjeh na predmetu.

Mentor:

Prof. dr. sc. Maja Matetić

Voditeljica za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 6.6.2019.

Mihaela Blašković
(potpis pristupnika)

Sadržaj

SAŽETAK

Problematika ovog rada je analiza podataka, zapisa studenata u sustavu za e-učenje. Objašnjeno je što su neuronske mreže, navedene su vrste mreža, detaljnije su objašnjene neuronske mreže s povratnom vezom te je predstavljen model za obradu prirodnog jezika BERT.

U radu su opisani sustavi za učenje, razlozi njihove ulazne popularnosti te su predstavljeni kao izvor velikog broja podataka za analizu sa svrhom unaprjeđenja obrazovanja. Upoznajemo se sa novim disciplinama čiji je fokus analiza podataka u edukaciji.

U istraživačkom dijelu je prikazana izrada modela za treniranje i testiranje podataka, predviđanje aktivnosti i klasifikaciju uspjeha ili neuspjeha.

Ključne riječi: analiza podataka, duboko učenje, neuronske mreže, sustavi za e-učenje

1 Uvod

U današnje vrijeme informacije su postale bitno „oružje“ poslovanja te prava informacija u pravo vrijeme može imati presudnu ulogu u svakodnevnom donošenju odluka ili rješavanju problema.

Potrebno je razlikovati pojmove podatak i informacija. Podatak je neka činjenica, činjenično stanje, dok je informacija podatak koji za primatelja ima određeno značenje. Kako bi podatak ili tok podataka postao informacija, potrebno je provesti analizu i otkriti njihovo značenje. Dubinska analiza podataka (*engl. data mining)* se može definirati kao proces prikupljanja, pretraživanja i analize veće količine podataka. Radi se na otkrivanju obrazaca, pravilnosti i odnosa u mnoštvu podataka, otkrivanju znanja u podacima koje u konačnici olakšava donošenje odluka. Fokus je na pronalasku relevantnih informacija i setova podataka kombinacijom alata i metoda korištenih u statistici i strojnom učenju. Analiza podataka može biti opisna ili prediktivna [1]. Ovaj rad se bavi prediktivnom analizom te će se u svrhu predikcije koristiti model umjetne neuronske mreže. Neuronske mreže su postigle značajne rezultate na područjima automatizacije zadataka poput generiranja teksta, prepoznavanja govora i problemu klasifikacije.

Područje kojim se bavimo je edukacija. U današnje vrijeme kada učenici nižih razreda koriste tehnologiju kao pomoćno sredstvo u nastavi, primjećuje se generalni porast digitalne transformacije kako u edukaciji tako i u ostalim područjima djelovanja. Time je došlo i do porasta generiranih podataka iz kojih bi se ispravnom upotrebom moglo doći do novih saznanja koja bi imala pozitivan učinak na procese u edukaciji za učenike i profesore.

Naredno poglavlje objašnjava motivaciju analize podataka na području edukacije te detaljnije opisuje sustave za e-učenje, LMS sustave, masivne online tečajeve te se upoznajemo sa novijim disciplinama u analizi podataka čiji je konačan cilj optimizacija procesa učenja.

Treće poglavlje je kratki prikaz strojnog učenja, potkategorije umjetne inteligencije čiji algoritmi se koriste u analizi podataka. Strojno učenje je set tehnologija koje mogu pomoći boljem razumijevanju podataka optimizacijom i prilagođavanjem metoda za učenje modela. U poglavlju su prikazani osnovni pristup učenju, nadzirano, nenadzirano, polunadzirano i učenje s podrškom.

Četvrto polje se bavi neuronskim mrežama. Neuronska mreža se može objasniti kao pokušaj oponašanja rada ljudskog mozga koji se temelji na korištenju slojeva međusobno povezanih

jedinica i veza temeljem iskustva. U poglavlju je prikazana osnovna jedinica neuronske mreže, neuron na primjeru perceptrona te modeli jednoslojnog i višeslojnog perceptrona.

Peto poglavlje se bavi neuronskom mrežom sa povratnom vezom i upoznavanjem sa LSTM arhitekturom.

U šestom poglavlju upoznajemo model BERT, noviji algoritam za obradu prirodnog jezika koji je osmislio i koristi Google.

Sedmo poglavlje se bavi analizom podataka gdje donosimo rezultate istraživanja.

2 Što je e-učenje?

Napredak tehnologije donosi poboljšanja na svim područjima ljudskog djelovanja, pa tako i na području obrazovanja. Spletom okolnosti nastalih pandemijom virusa COVID-19 i obrazovanje kakvog ga poznajemo kao tradicionalni način učenja sa profesorom pred pločom i učenicima u klupama prolazi fazu digitalne transformacije. Tim procesom se velik dio sudionika obrazovanja susreće sa novinama koje je potrebno savladati uz već postojeće sadržaje određene kurikulumom. Nadalje, obrazovanje ima bitnu ulogu u procesu oblikovanja i pripremi mladih naraštaja za svijet koji se danas sve više oslanja na tehnologiju i mogućnosti koje pruža. Stoga se potiče primjena digitalne tehnologije u procesu te razvoj digitalnih kompetencija i vještina. Na taj način, korištenjem tehnologije u obrazovanju provodi se e-učenje.

Termin „e-učenje“ se odnosi na sustav učenja pomoću tehnologije za pristup obrazovnom programu izvan tradicionalne učionice u bilo kojem trenutku i bilo gdje [2]. Svoje začetke pronalazi u učenju, obrazovanju na daljinu kada su tečajevi bili organizirani u obliku slanja materijala polaznicima tečaja putem pošte. Student je naručio materijale, proučio ih i vraćao zadane mu zadatke poštom. Na kraju tečaja, ispit se polagao na isti način, šaljući rezultate poštom koje je profesor ocjenjivao. Godine 1858. Sveučilište u Londonu je postalo prvo sveučilište koje je omogućilo stjecanje diplome putem obrazovanja na daljinu čime je započeo razvitak tzv. dopisnih studija [3].

E-učenje kao koncept ne predstavlja novu pojavu, već je novost tehnologija koja je to omogućila. Današnje e-učenje u kojem mogu sudjelovati svi naraštaji koji imaju pristup internetu, provodi se gledajući i videozapise na YouTube platformi gdje ne samo profesori, već mnogi stručnjaci iz različitih područja djelatnosti objavljuju videozapise u kojima dijele svoje znanje. Čitanjem stručnih časopisa, sudjelovanjem u raspravama na forumima, pristupanjem tečajevima na nekim od mnogobrojnih platformi poput primjerice Udemy, DataCamp ili Skillshare mogu se naučiti nove vještine i proširiti postojeće znanje.

Jedna od takvih platformi otvorenog koda koja omogućuje e-učenje i s kojom su upoznati i studenti Sveučilišta u Rijeci je Moodle što je skraćenica za Modularno objektno-orijentirano dinamičko obrazovno okruženje (*engl. Modular Object-Oriented Dynamic Learning Environment*). Radi se o jednom od najčešće korištenih LMS sustava (*engl. Learning Management System*) koji omogućuje online suradnju između profesora i studenata. Više o LMS sustavima bit će govora u sljedećem poglavlju.

2.1 LMS ili sustavi za upravljanje učenjem

LMS ili sustav za upravljanje učenjem je aplikacija ili web bazirana tehnologija koja se koristi za planiranje, implementaciju i omogućavanje pristupa procesu učenju. Sustav pruža mogućnosti kreiranja i upravljanja sadržajem, pomaže u administraciji, kreiranju rasporeda i rokova, kreiranju provjera znanja, praćenju i izvještavanju o napretku pojedinog polaznika te omogućava interakciju između korisnika.

Osnovne uloge u sustavu su administratori, predavači i studenti. Administrator sustava upravlja tečajevima i korisnicima, može ih dodavati i brisati te je podrška ostalim korisnicima sustava. Predavači kreiraju sadržaj na tečaju koji otvara administrator te obavljaju ostale nastavne funkcije poput izrade provjera znanja i diskusija.

Neke od prednosti sustava su organizacija i sigurno spremanje podataka pomoću sigurnosnih kopija (*engl. back-up*) i šifriranja (*engl. data encryption)*, praćenje izvedbe i napretka polaznika, personalizacija učenja prema potrebama polaznika, dostupnost materijala i aktivnosti u svako vrijeme i mjesto.

Uz osnovne funkcionalnosti LMS sustavi mogu pružati različite funkcionalnosti pomoću kojih se želi poboljšati kvaliteta učenja. Neke od funkcionalnosti mogu se odnositi na razne integracije, način rada, certificiranje, audio/video alate, tzv. *gamification* gdje se pomoću elemenata i principa igre pokušava motivirati polaznike.

Istraživanja su pokazala da [1]:

- Ustanove i studenti cijene LMS sustave kao poboljšanje iskustva u učenju, ali relativno mali broj iskorištava pun kapacitet sustava,
- Zadovoljstvo korisnika je najveće za osnovne LMS funkcionalnosti, dok je najniže za mogućnosti suradnje,
- Ustanove napominju kako bi učitelji bili efektivniji, a studenti uspješniji uz vještije korištenje sustava,
- Obje strane se slažu u želji za poboljšanim i personaliziranim funkcionalnostima i korištenju analitike u poboljšanju učenja.

LMS sustavi se često koriste i u korporativnom svijetu za obrazovanje zaposlenika tvrtke, no najpopularniji LMS sustav koji je u proteklih 18 godina skupio preko 223 milijuna korisnika i28 milijuna tečajeva je upravo Moodle. Radi se o platformi koja se najčešće koristi u

visokoškolskim ustanovama (67%) gdje svaka ustanova ima inačicu prilagođenu svojim potrebama. Autor Moodle platforme je Martin Dougiamas. Prvi prototip je kreiran 1999., dok je prva službena verzija Moodle 1.0 isporučena u kolovozu 2002. godine [4].

Moodle je besplatni LMS otvorenog koda (*engl. open source*) koji pruža prilagodljivo okruženje za učenje te sustav iza kojeg stoji velika zajednica korisnika. Primjeri ostalih LMS sustava su TalentLMS, Blackboard, Schoology, Docebo. Još jedan poznati primjer sustava korišten na Sveučilištu u Rijeci uz već poznati Moodle je Canvas. U sljedećem poglavlju će se pobliže objasniti inačica Moodle-a „Merlin".

2.2 Merlin

„Sustav za e-učenje Merlin omogućava nastavnicima, studentima i ustanovama u sustavu visokog obrazovanja izvođenje kolegija, koji se nalaze u redu predavanja pojedine ustanove, uz primjenu tehnologija e-učenja" [5].

Ovaj diplomski rad se bavi analizom podataka prikupljenih sa sustava za e-učenje kojim se koriste zaposlenici i studenti Sveučilišta u Rijeci, tzv. Merlin sustav za učenje. Merlin je inačica Moodle platforme. U slučaju Merlina, radi se o platformi na koju profesori i asistenti postavljaju materijale za učenje, prenose obavijesti studentima, kreiraju online ispite, vode evidenciju bodova i ocjena. Također, studenti preuzimaju materijale za učenje, rješavaju provjere, imaju mogućnost međusobnog razgovora putem chata i komunikacije s profesorima putem foruma i poruka te mogu pohranjivati svoje materijale i predavati na ocjenjivanje zadaće i ispite. Sustav je za mnoge kolegije tradicionalne alate edukacije poput bilježnica, indeksa i imenika učinio nepotrebnim te i profesorima i studentima pojednostavio proces učenja i digitalizirao papirologiju.

Sustav za učenje Merlin također ima velik broj korisnika, brojke za akademsku godinu 2019/2020 su sljedeće: 83.565+ studenata, 9.213+ nastavnika i preko 23.716 kolegija. Podaci koji se koriste u izradi rada su iz 2017/2018 kad je sustav brojao nešto više od 10.000 kolegija i sveukupno 58.966 korisnika [6]. Ipak, te brojke su još uvijek manje od sličnih sustava obrazovanja tzv. MOOC koji su nastali na ideji „demokratizacije" sadržaja elitnih sveučilišta.

2.3 MOOC ili masivni javni online tečajevi

E-učenju se može pristupiti i putem MOOCs (*engl. Massive Open Online Course*) platformi poput već navedene Udemy. MOOC se može definirati kao tehnološka inovacija, odnosno jeftina alternativa tradicionalnom i skupom načinu učenja, online tečaj usmjeren prema većem broju sudionika i dostupan putem interneta. Popularnosti online tečajeva je doprinijela i činjenica da je tako omogućen pristup informacijama, odnosno procesu učenja osobama diljem svijeta. Drugim riječima, osobe iz raznih zemalja mogu u isto vrijeme pohađati tečaj koji se inače održava primjerice na Harvardu. Mnoge ugledne institucije pružaju usluge online tečaja na nekoj od platformi, a neke od njih su: Coursera, edX, Udacity i Futurelearn. Lekcije prestižnih fakulteta su odjednom nadohvat ruke i privlače široku masu, posebice one koji žele naučiti nešto novo što bi im koristilo na poslu, usavršavati se i steći certifikat za neku od vještina. Termin „MOOC“ je nastao 2008. kojim se definirao prvi MOOC tečaj "Connectivism and Connective Knowledge" koji su pohađali studenti na Sveučilištu Manitoba te velik broj studenata koji su besplatno pristupili tečaju online. U današnje vrijeme, broj korisnika na MOOC platformama se mjeri u milijunima [1], [7].

Upravo ta velika brojka je značajna za analizu podataka u području edukacije. Sustavi za e-učenje generiraju velik broj podataka te kako je riječ o studentima diljem svijeta, populaciji koja se razlikuje na kulturnoj i demografskoj razini, skup podataka koji se može prikupiti je veoma heterogen (spol, rasa, vjera, stopa prolongiranja, odustajanja, utrošeno vrijeme na videozapise ili prezentacije i sl.) [1].

Zaključak je da postoje dvije velike prednosti masivnih online tečajeva, a to su velik skup podataka koji je ujedno i heterogen. Mnogi polaznici samo započnu tečaj, ne završe ga, odustanu, no svejedno sustavi prikupljanju takve i ostale podatke koji u konačnici pomnim analizama mogu dovesti do novih informacija. Na temelju takvih podataka mogu se provoditi analize o tome koji je bolji način učenja za studente, da li su to lekcije u natuknicama, video lekcije ili kratke učestale samoprovjere? Hoće li studenti ostvariti bolji uspjeh ako ih se bombardira svako malo nekom kratkom provjerom ili su dovoljni veliki ispiti? Analizom kvalitetnog skupa podataka mogu se dobiti odgovori na takva pitanja čime se u konačnici može poboljšati sustav obrazovanja.

Obrazovanje u tradicionalnoj učionici ili na online kolegiju na kojem je upisano 50 studenata ili na kolegij na kojem je upisano 1.000 studenata se ne može provoditi na isti način. Obrazovanje u tradicionalnoj učionici ima svoje specifične karakteristike, ali postoje i razlike

u online kolegijima sa 100 ili 10.000 studenta i stoga je bitno razviti tehnike kojima će svima biti pruženo ista razina kvalitete obrazovanja. MOOC sustavi su otvorili mnogobrojne prilike u obrazovanju za tradicionalno i netradicionalno učenje i upravo želja za napretkom i otkrivanjem novih tehnika su motivacija za analizu podataka u području edukacije.

2.4 Razlike između Moodle i MOOC platformi

Moodle je sustav za upravljanje učenjem, LMS, dok je MOOC online tečaj usmjeren ka velikom broju polaznika. Ispravnija bi bila usporedba LMS sustava i MOOC platforme, no u ovom slučaju je fokus na podacima iz poznatog sustava i stoga će se koristiti u usporedbi.

Postoji nekoliko razlika između Moodle i MOOC platformi i u ovom poglavlju su navedene najbitnije. Moodle je pomoćno sredstvo u edukaciji u kojima je još uvijek prisutan i tradicionalni način učenja poput održavanja predavanja u učionicama, pisanje ispitnih provjera na papiru, no u novije doba i to se gubi i sve postaje digitalno. Također, Moodle se u većini slučajeva koristi u okviru institucije, najčešće na fakultetu gdje postoji školski kurikulum, određene su obaveze profesora i studenata, definirani rokovi koji se moraju poštovati, dan je vremenski rok u kojem je potrebno položiti kolegij jer se u suprotnom slučaju kolegij pada i mora se ponovno odslušati. Treba napomenuti da su prvi MOOC tečajevi koristili Moodle te pravilno konfiguriran i Moodle može podržati MOOC tečaj. Najveća registrirana Moodle stranica trenutno ima preko milijun korisnika [4].

Na MOOC platformi, svaki polaznik je zaseban, samovoljno pristupa tečaju, ima pristup materijalima i može ih proučavati u bilo koje doba te isto tako ako postoji, polaže ispit u doba koje mu odgovara. Polazniku nisu nametnuti rokovi, sam određuje svoj tempo učenja i odrađivanja obaveza. No, to isto rezultira manjim brojem završetka tečajeva.

Postoji razlika i u broju polaznika. Drugim riječima, na MOOC platformama, kao što ime kaže „masovni“ tečajevi, prijavljuju se milijuni potencijalnih studenata koje većina Moodle inačica ne bi mogla tehnički podnijeti. Iz tog razloga pristupačnosti i povezivanja otvorenih tečajeva sa prestižnim sveučilištima su MOOC platforme dobile na popularnosti te je New York Times 2012-tu godinu proglasio „godinom MOOC-a“ [1].

Zaključno, neovisno radi li se o Moodle platformi ili MOOC tečaju, e-učenje rezultira velikim skupovima podataka obrazovnog konteksta koje nije moguće ručno analizirati. Postavlja se

pitanje kako iskoristiti postojeće podatke. Pojavila se potreba za alatima koji bi automatski analizirali podatke i dali informacije koje mogu koristiti svi sudionici obrazovanja.

2.5 Dubinska analiza podataka u obrazovanju

Generiranje velikih količina podataka različitih izvora i formata je rezultiralo novim disciplinama koje se bave isključivo analizom podataka s područja edukacije, tzv. Dubinskom analizom podataka u obrazovanju *(engl. Educational Data Mining, EDM)*. Disciplina se bavi razvojem metoda za istraživanje podataka iz edukacijskih okruženja. Definira se kao primjena metoda dubinske analize podataka pomoću kojih se rješavaju važna pitanja iz obrazovanja. Uz EDM treba spomenuti i Analitiku učenja *(engl. Learning Analytics, LA)*, disciplinu sa fokusom na mjerenju, prikupljanju, analizi i izvještavanju o studentima u svrhu razumijevanja i optimizacije učenja. LA se fokusira na izazove u obrazovanju, integriranje tehnologije u socijalne, pedagoške dimenzije učenja poput prepoznavanje studenata za koje je rizik napuštanja tečaja veći ili identifikacija studenata koji imaju poteškoće tijekom tečaja. Nasuprot tome, EDM se bavi tehničkim izazovima, primjerice traži nove uzorke u podacima i pokušava razviti nove algoritme. Unatoč razlikama, radi se o interdisciplinarnim područjima koja se značajno preklapaju u metodama i tehnikama koje koriste te dijele zajednički interes: analizu generiranog toka podataka nastalih tijekom interakcije studenata sa različitim tehnologijama učenja kako bi se razumio proces učenja i izvukla povratna informacija [8].

Porast korištenja tehnologije u obrazovanju je doveo do većeg broja podataka za analizu i institucije se suočavaju sa eksponencijalnim rastom podataka i njihovom transformacijom u informacije na korist svima. Ti podaci mogu biti veoma različiti, od osnovnih informacija o studentu poput mjesta i datuma rođenja, zanimanja roditelja do broja pristupa određenom modulu, datoteci ili vremena koje je proveo odgovarajući na pitanje u online kvizu.

Podaci o provedenim aktivnostima u LMS sustavu se spremaju u dnevničke zapise, tzv. logove koji pružaju uvid u ponašanje studenata. Logovi su lista događaja gdje svaki zapis ili linija sadrži vremenski žig (*engl. time-stamp*) i dodatne informacije o aktivnosti. Svaki klik u sustavu se pohranjuje u listu logova. Iz logova je moguće saznati koje sve aktivnosti je napravio korisnik. Moodle log datoteka se sastoji od datuma i vremena kad je aktivnost odrađena, imena korisnika, informacije o aktivnostima po modulima te koja je točno akcija provedena i IP adrese sa koje student pristupa. Tako se za svakog korisnika sustava kreira vremenski niz njegovih pristupa sustavu.

Takvi podaci uz odrađene transformacije mogu se koristiti za analizu podataka. Uzimajući u obzir izvor i karakteristike podataka, najčešće metode strojnog učenja koje se koriste u analizi podataka na području edukacije su: stablo odlučivanja, umjetne neuronske mreže, logistička regresija, Bayesova statistika [1]. Za potrebe rada je korištena umjetna neuronska mreža.

3 Strojno učenje

Strojno učenje tzv. ML (*engl. machine learning*) je znanost programiranja računala tako da mogu učiti iz podataka. Radi se o grani umjetne inteligencije. Disciplina se bavi proučavanjem algoritama čija učinkovitost raste kroz iskustvo. Može se definirati na sljedeći način [9]:

„A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.“

– T. Mitchell, 1997.

Jedna od prvih primjena strojnog učenja većih razmjera je „spam“ filter 1990-ih. Spam filter je program strojnog učenja koji je na temelju primjera spam mailova (koji su označeni oznakom „spam“ od strane korisnika) i standardnih mailova sposoban naučiti kako prepoznati novi spam. Primjeri na kojima se sustav uči je set za učenje ili treniranje i svaki primjer predstavlja jednu instancu ili uzorak. U slučaju prepoznavanja spama, zadatak T je označavanje oznakom spam novih mailova, iskustvo E je treniranje skupa podataka i mjera uspješnosti P može biti omjer ispravno klasificiranih mailova. Kad je spam filter istreniran na dovoljnom broju spam mailova, može se napraviti revizija liste riječi i kombinacija riječi za koje se pretpostavlja da su najčešći pokazatelji spama. Često takve liste otkrivaju neočekivane korelacije ili nove trendove čime se dolazi do boljeg razumijevanja problema. Primjena metoda strojnog učenja na veće količine podataka omogućava otkrivanje obrazaca koji nisu odmah vidljivi. Taj proces se naziva dubinska analiza podataka [9].

Strojno učenje se koristi za pojednostavljenje postojećeg koda, u kompleksnijim problemima kada tradicionalni pristup ne donosi najbolja rješenja, kada je potrebno sustav prilagoditi novim podacima. Svakodnevni problemi koji se uz pomoć ML uspješno rješavaju su ručni unos podataka, detekcija spama, sustav preporuka, medicinska dijagnostika i mnogi drugi.

Osnovni pristupi strojnom učenju su nadzirano učenje, nenadzirano, hibridno ili polunadzirano učenje i podržano učenje ili učenje podrškom.

Nadzirano učenje (*engl. supervised learning*) je učenje u kojem su poznati parovi ulaznih i izlaznih vrijednosti, a cilj je naučiti model kako predvidjeti izlazne vrijednosti iz danih ulaznih. Najčešći zadaci nadziranog učenja su klasifikacija (predviđanje klasa) i regresija (predviđanje vrijednosti). Neki od algoritama nadziranog učenja su algoritam k-najbližih susjeda, linearna

regresija, logistička regresija, stroj s potpornim vektorima (SVM), stablo odluke, neuronske mreže.

Nenadzirano učenje (*engl. unsupervised learning*) je strojno učenje bez „učitelja“ u kojem je poznat samo ulazni skup podataka te je potrebno otkriti pravilnosti između podataka. Neki od najčešćih algoritama nenadziranog učenja su grupiranje (*engl. clustering*), vizualizacija podataka, redukcija dimenzionalnosti i asocijativno učenje.

Hibridno ili polunadzirano učenje (*engl. semisupervised learning*) je strojno učenje gdje algoritmi rade sa skupom za učenje koji sadrži djelomično označene i veće količine neoznačenih podataka. Radi se o kombinacijama algoritama nadziranog i nenadziranog učenja. Primjerice duboke mreže vjerovanja (*engl. deep belief network, DBNs*) se temelje na nenadziranim komponentama ograničenog Boltzmanovog stroja (*engl. RBMSs*) naslaganim u stog. RBMSs se trenira iterativno nenadziranim načinom, a zatim se cijeli sustav podešava pomoću tehnika nadziranog učenja.

Podržano učenje ili učenje s podrškom *(engl. reinforcement learning*) je učenje gdje postoji tzv. softverski agent koji promatra okruženje, odabire i izvodi radnje te zauzvrat dobiva nagrade ili kazne. Agent mora sam naučiti koja je najbolja strategija za dobivanje najviše nagrada na temelju čega se odlučuje koju radnju odabrati u danoj situaciji. Učenje se temelji na pokušajima i pogreškama sa ciljem maksimalnog uspjeha i minimalnog neuspjeha. Ovaj pristup se koristi kada je poznat izlaz koji se želi dobiti. Podržano učenje koriste roboti prilikom učenja hodanja [9].

Strojno učenje je disciplina koja se bavi kreiranjem i istraživanje algoritama koji se kroz iteracije poboljšavaju. Kao odgovor na neke nedostatke strojnog učenja i obzirom na napredak u teoretskim i tehnološkim mogućnostima, znanost dubokog učenja postaje sve popularnija. Koristi se u tehnologijama poput autonomnih vozila, prepoznavanju slika i slično. Duboko učenje je potkategorija strojnog učenja usmjerena na kreiranje algoritama koji objašnjavanju i uče visoku i nisku razinu apstrakcije podataka. Modeli dubokog učenja su često inspirirani drugim izvorima znanja poput strukture ljudskog živčanog sustava. Izraz „duboko“ se odnosi na serije koraka ili slojeva čijim povećavanjem je moguća obrada većih količina i kompleksnosti podataka. Okosnica dubokog učenja su neuronske mreže [10].

4 Neuronske mreže

4.1 Osnove neuronskih mreža

Neuronska mreža je metoda strojnog učenja nadahnuta procesom obrade informacija u živčanom sustavu. Začecima neuronskih mreža smatra se 1943. godina kada su Warren McCulloch i Walter Pitts u radu „*A Logical Calculus of Ideas Immanent in Nervous Activity*“ predstavili pojednostavljeni model rada biološkog neurona koji je kasnije postao poznati kao „umjetni neuron“.

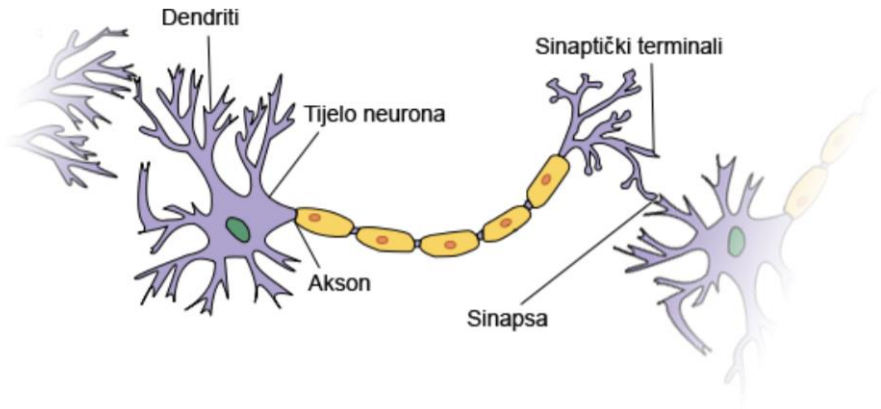
Neuronska mreža se može shvatiti kao model paralelnog rada jednostavnih jedinica. Jednostavna jedinica je neuron ili živčana stanica, osnovna jedinica živčanog sustava. Informacije u mreži se dugotrajno spremaju u tzv. 'težine' između jedinica, neurona te je ažuriranje težina primarni način kako neuronska mreža uči nove informacije [11].

Obrada informacija u živčanom sustavu se vrši pomoću neurona. Neuroni u živčanom sustavu međusobno komuniciraju, prosljeđuju elektrokemijske impulse od jednog prema dugom sve dokle je impuls iznad minimalnog praga aktiviranja (*engl. treshold*).

4.1.1 Biološki neuron

Osnovni dijelovi biološkog neurona su soma ili tijelo neurona, dendriti, akson i sinapse. Tijelo neurona može imati mnogo izdanaka dendrita i samo jedan produžetak akson. Na kraju aksona se nalaze sinaptički terminali ili jednostavnije, sinapsa koja je zatim povezana na dendrite drugog neurona ili direktno na tijelo neurona.

Neuroni su međusobno povezani putem sinapsa preko kojih se prenose impulsi odnosno informacije od jednog neurona drugome. Dio neurona koji prima signale od drugih neurona su „antene neurona“, dendriti. Signali mogu biti pozitivni ili negativni. Svi signali koji stižu u neuron se sumiraju i ako je konačna suma veća od granične vrijednosti, neuron generira izlazni signal prema drugome neuronu preko aksona. Signali prvo preko aksona stižu do sinapsa i zatim sinapse šalju signale dendritima drugih neurona čime je omogućen prijenos informacija [9] [12].



Slika 1: Biološki neuron [13]

Matematički, neuronska mreža pokušava kreirati funkciju koja će usmjeriti ulazne vrijednosti prema željenim izlaznim vrijednostima. Neka od područja gdje se neuronske mreže na taj način primjenjuju su: klasifikacija i kategorizacija teksta, prepoznavanje govora, predviđanja i analiza financijskog tržišta kao i vremenskih prilika, otkrivanje eksploziva u prtljazi, istraživanje ležišta nafte, računalni vid.

4.1.2 Umjetni neuron

Na temelju biološkog neurona, nastala je ideja umjetnog neurona (*engl. artificial neuron*) na temelju spomenutog rada „*A Logical Calculus of Ideas Immanent in Nervous Activity*“. Umjetni neuron je model sa jednim ili više binarnih ulaza i jednim izlazom. Izlaz se aktivira kad je određen broj ulaza aktivan. Na temelju tog pojednostavljenog modela moguće je izgraditi neuronsku mrežu koja može obavljati jednostavne logičke operacije [9].

U radu se primjenjuje umjetna neuronska mreža tzv. ANN (*engl. Artificial Neural Network*). Osnovna struktura ANN mreže se sastoji od malih povezanih procesnih jedinica po uzoru na biološku neuronsku mrežu gdje svaki neuron obavlja operaciju generiranja jednog izlaza iz skupa ulaza [14]. Pobliže će biti objašnjena u narednom poglavlju.

4.2 Perceptron

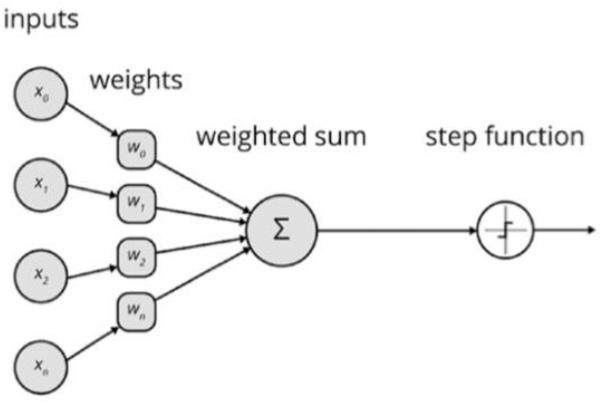
Perceptron je kreiran kao pojednostavljeni matematički model neurona, jedna od najjednostavnijih ANN arhitektura. Temelji se na umjetnom neuronu tzv. LTU (*engl. linear treshold unit).* Radi se o konceptu u kojem su vrijednosti numeričke i svaki unos/ulazna vrijednost ($x_1, x_2, \ldots, x_n$) je određena težinom ($w_1, w_2, \ldots, w_n$) koja je numerički izraz jakosti veza.

Na zbroj umnoška ulaznih vrijednosti i pripadajućih težina ($z = w_1x_1 + w_2 x_2 + \ldots + w_nx_n = w^T * x$) se primjenjuje tzv. funkcija koraka (*engl. step function*) i generira izlaz $h_w(x) = step\ (z) = step\ (w^T * x)$. Još neke od korištenih aktivacijskih (prijenosnih ili transfer) funkcija su funkcija identiteta, logistička sigmoidalna, tanges hiperbolni funkcija, softmax, ReLu.

Perceptron se može koristiti za probleme linearne binarne klasifikacije. Izračunava linearnu kombinaciju ulaznih vrijednosti i ako rezultat premašuje definirani prag, izlazna vrijednost će biti pozitivna, +1, a u suprotnome je -1. Vrijednost praga je indikator kada neuron postaje aktivan i prema tome možemo tu vrijednost shvatiti kao parametar aktivacijske funkcije. Često se u mrežu uključuje dodatni, tzv. bias neuron čiji je izlaz 1 i služi kao pomak aktivacijske funkcije, ima djelovanje samo na izlaz [12], [9].

Perceptron ne posjeduje prethodno znanje, stoga su početne težine nasumične. Ukoliko rezultat nije zadovoljavajući, sinaptičke težine se ažuriraju do željenog rezultata, odnosno smanjenja greške. Greška modela se smanjuje ili povećava ovisno o ažuriranju težine [10].

Više perceptrona se mogu povezati u mrežu čime nastaje mreža jednoslojnog perceptrona tzv. SLP (*engl. Single Layer Perceptron*). Radi se o modelu u kojem postoji jedan ulazni sloj i jedan izlazni sloj. Iako se čini kao da su to dva sloja, ulazni sloj se ne broji jer nije sloj u kojem se odvijaju neki procesi već samo sadrži ulazne vrijednosti i stoga je naziv „jednoslojni“. SLP mreža je unaprijedna neuronska mreža što znači da su neuroni ulaznog sloja povezani s neuronima izlaznog sloja, nema poveznica između neurona unutar istog sloja [12]. Na slici ispod je prikazana mreža jednoslojnog perceptrona s četiri ulazna neurona i jednim izlaznim.



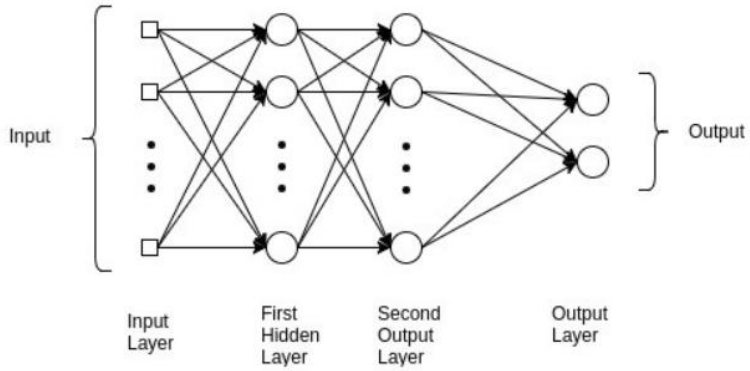
Slika 2: Model jednoslojnog perceptrona [10]

Osnovni model neuronske mreže je model jednoslojnog perceptrona. SLP modeli su ograničeni utoliko što su točni samo u radu s podacima koji su linearno odvojivi. Isto se javlja kao problem

u radu sa složenijim podacima i time je ovaj model poslužio kao inspiracija za razvoj ostalih vrsta neuronskih mreža i modela strojnog učenja u rješavanju kompleksnijih problema [10].

4.3 Višeslojni perceptron

Kao odgovor na rješavanje kompleksnijih problema model neuronske mreže koji se razvio iz jednoslojnog perceptrona je višeslojni perceptron ili MLP (*engl. Multi-Layer Perceptron*). Razlika od jednoslojnog perceptrona jest da u modelu višeslojnog perceptrona postoji skriveni sloj (*engl. hidden*) koji utječe na izlaz, odnosno rezultat modela. MLP model se sastoji od jednog ulaznog sloja, jednog ili više skrivenih slojeva i naposljetku se nalazi jedan izlazni sloj. Slojevi su potpuno povezani što znači da je svaki neuron u prvom sloju povezan sa svim ulazima. Isto tako svaki neuron iz sljedećeg sloja je povezan na sve neurone prethodnog sloja, tj. Imamo unakrsno povezivanje gdje je svaki neuron povezan sa svim ostalima kao što je prikazano na slici ispod [12], [9].



Slika 3: Višeslojni perceptron [15]

Ako ANN mreža ima dva ili više skrivenih slojeva, radi se o „DNN“, dubokoj neuronskoj mreži (*engl. deep neural network*) [9].

Višeslojni perceptron je najčešće korištena unaprijedna neuronska mreža, FNN (*engl. feedforward neural network*). Ulazne vrijednosti se spremaju u ulazni sloj odakle prolaze kroz skrivene slojeve do izlaznog sloja. Ovaj proces je poznat kao „prolaz unaprijed“ (*engl. forward pass*) [14].

Za učenje MLP-a i ostalih neuronskih mreža se standardno koristi algoritam unazadne propagacije (*engl. back-propagation*). Konačni cilj algoritma je optimizacija sinaptičkih težina

na temelju stope pogreške dobivene u prethodnoj iteraciji. Pravilna prilagodba težina osigurava nižu stopu pogreške što čini model neuronske mreže pouzdanijim. Funkcija koja se koristi za izračun stope pogreške učenja je tzv. funkcija gubitka (*engl. loss function*). Stopa pogreške pokazuje kolika je razlika, točnije grešku između dobivenih i željenih rezultata. Kako bi se izvršila optimizacija potrebno je znati u kojem smjeru djelovati i za to se često koristi funkcija gradijentnog pada (*engl. gradient descent*). Gradijentni pad mjeri koliko će se promijeniti izlaz funkcije ako se promjene ulazi. Pomoću njega se određuje smjer greške i sinaptičke težine koje je potrebno ažurirati da bi se smanjila stopa greške i unaprijedila neuronska mreža [16]. Princip rada je da prilikom svake instance učenja, algoritam unazadne propagacije prvo radi predviđanje, mjeri grešku učenja, a zatim prolazi kroz svaki sloj mreže unatrag mjereći doprinos pogreške iz svake veze što predstavlja izračun gubitka te u konačnici podešava težine veza kako bi se smanjila greška [9].

Budući da izlaz mreže višeslojnog perceptrona ovisi samo o trenutnom ulazu, MLP mreže se najčešće koriste u problemima klasifikacije i regresije, npr. klasificiranje da li je nešto hitno ili nije, spam ili uobičajena dolazna pošta [14].

4.4 Podjela umjetnih neuronskih mreža

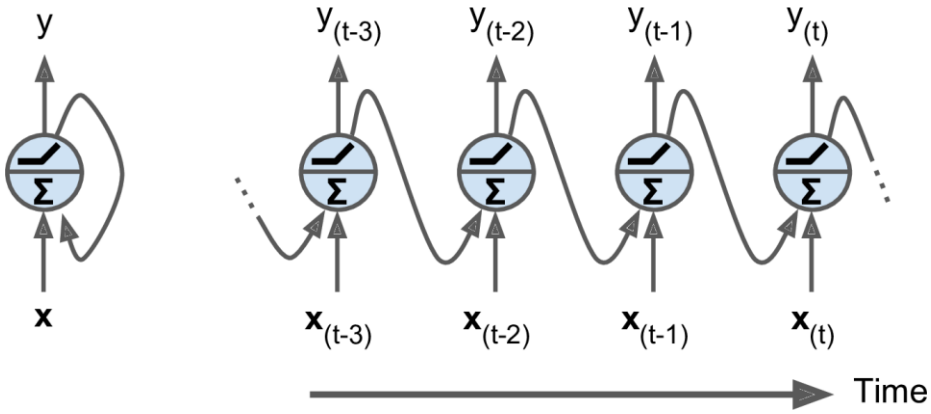
Ovisno o arhitekturi mreže, osnovna podjela neuronskih mreža jest na :

- Cikličke – sadrže povratne veze, tzv. mreže s povratnom vezom,
- Acikličke – ne sadrže povratne veze, informacije imaju ravan pravac putovanja.

U acikličkim mrežama signali putuju od ulaznog sloja preko skrivenog u izlazni i stoga se još nazivaju unaprijednim neuronskim mrežama, tzv. FNN koje smo upoznali na primjerima perceptrona. Cikličke mreže će biti detaljnije objašnjene u sljedećem poglavlju na primjeru neuronskih mreža sa povratnom vezom.

5 Neuronske mreže s povratnom vezom ili RNN

Kao što je prikazano u prethodnim poglavljima, neuronska mreža nema memorije, svaki ulaz se individualno obrađuje bez ovisnosti o ostalim ulazima. Podaci koji se obrađuju u diplomskom radu su sekvencijalni podaci, s određenim poretkom, tj. vremenskom ovisnosti. Za obradu takvih podataka potrebna je neuronska mreža s povratnom vezom ili povratna neuronska mreža, kraće RNN (*engl. Recurrent Neural Network*). Slična je unaprijednoj neuronskoj mreži, osim što sadrži povratne veze. Nastale su kao pomoć neuronskim mrežama u radu s kompleksnim zadacima analiziranja sekvencijalnih podataka.



Slika 4: Neuronska mreža s povratnom vezom koja se može „odmotati" kroz vrijeme [9]

Bitna odrednica RNN je petlja koja omogućava mreži da prolazi kroz sekvencijalne ulazne podatke dok istovremeno zadržava unutarnje, tzv. stanje ćelije između koraka.

Slika 4 prikazuje najjednostavniju RNN mrežu s jednim neuronom. Na lijevoj strani slike je prikazana osnovna neuronska mreža s ulazom, jednim neuronom u skrivenom sloju i izlazom. Na desnoj strani se nalazi isti prikaz, samo što je prikazano procesiranje petlje kroz vrijeme, „odmotana" reprezentacija mreže na desnoj strani. Primjerice, u određenom trenutku *t-1*, neuron prima informaciju i izračunava izlaz na temelju izlaza prethodnog koraka, $y_{(t-2)}$ te trenutnog ulaza mreže $x_{(t-1)}$. „Odmotana" RNN se može gledati kao lanac identičnih unaprijednih mreža [9].

U unaprijednoj neuronskoj mreži informacije imaju strog pravac putovanja, od ulaza kroz skrivene slojeve do izlaza. Novost koju donosi RNN jest da izlaz trenutnog vremenskog koraka postaje ulaz sljedećeg koraka. Kod svakog elementa vremenske serije, mreža uzima u obzir trenutni ulaz i informaciju o prethodnom stanju [17]. Drugim riječima, u neuronu se povezuje ulaz trenutnog koraka i memorija iz prethodne iteracije nakon čega se rezultat prosljeđuje

aktivacijskoj funkciji. Aktivacijska funkcija standardno šalje izlaz van i putem radne memorije u sljedeću iteraciju.

5.1 Učenje neuronske mreže s povratnom vezom

„Obična“ neuronska mreža koristi algoritam unazadne propagacije za ažuriranje težina i smanjenje greške u razlici između očekivanog i predviđenog izlaza te upotrebljava gradijentni spust za pronalazak optimalne vrijednosti težina.

Ciklička priroda RNN-a podrazumijeva da svaka petlja ima svoj par ulaz-izlaz iako sve zajedno dijele iste težine što predstavlja problem prilikom ažuriranja iste. Problem je riješen uvođenjem algoritma BPTT (*engl. Backpropagation Through Time*) ili algoritam unazadne propagacije kroz vrijeme. BPTT se aktivira nakon unaprijedne propagacije i odrađuje ažuriranje težina na sljedeći način:

- Mreža se prvo odmota u lanac unaprijedne mreže. Svaka ANN se može promatrati kao kopija originalnog RNN-a dijeleći iste težine i aktivacijsku funkciju,
- BPTT djeluje unatrag kroz lanac, računajući gubitak i greške u svakoj ANN,
- Mreža se „smota“ u prvobitno stanje i ažuriraju se težine.

Ograničenja RNN-a:

RNN koriste tzv. „radnu memoriju“ za pamćenje stanja iz prethodnih iteracija, no nije učinkovita u pamćenju mnogo ranijih stanja, odnosno dužih vremenskih zavisnosti. Uzrok zaboravu je poznat kao problem nestajućeg gradijenta. BPTT u više vremenskih koraka ima isti učinak neprestanog smanjenja gradijenta pogreške što otežava spuštanje gradijenta i optimizaciju težine.

Neki od primjera gdje se danas koriste RNN mreže su npr. :

- autonomni sustavi za vožnju – zbog rastresenih vozača koji prelaze preko traka, pješaka koji iskaču između parkiranih automobila, djece koje se igraju pokraj ceste ili životinje koje prelaze cestu, vožnja automobila može postati vrlo opasna i nepredvidljiva. U autoškoli polaznici uče kako uvijek moraju očekivati opasnu situaciju i biti spremni

reagirati, no nažalost često dođe do zakašnjelih reakcija. Autonomna vozila bi trebao voziti bez uplitanja vozača. Trenutno vozila koriste senzorne podatke poput niza slika za prepoznavanje objekata u pokretu te RNN mreža analizira podatke i predviđa budući položaj i brzinu objekta. Predviđanje budućih kretnji objekata pomaže vozaču u pravovremenom donošenju odluka tijekom vožnje i samo sustavu informacije za automatsko kočenje [18],

- obrada prirodnog jezika ili NLP (*engl. Natural Language Processing*) – jedna od najpoznatijih primjena RNN na području NLP-a je strojno prevođenje gdje se rješava problem prevođenja teksta iz primjerice engleskog u hrvatski. Još neki od primjera NLP su generiranje opisa slike ili videozapisa, transformacija govora u tekst, generiranje sažetka većeg dokumenta [17],

te u ostalim sličnim zadacima u kojima je potrebno donositi odluke temeljem prethodnih informacija. Jedna od prednosti RNN mreže jest što mogu raditi s podacima različitih dužina. Također, što je veća količina podataka, točnost modela će biti veća. Primjerice, ako se predviđa količina oborina, model će biti točniji ako postoje podaci od prije 10 godine u odnosu na podatke od samo 1 godine.

Nastavno na RNN su nastale sljedeće arhitekture [10]:

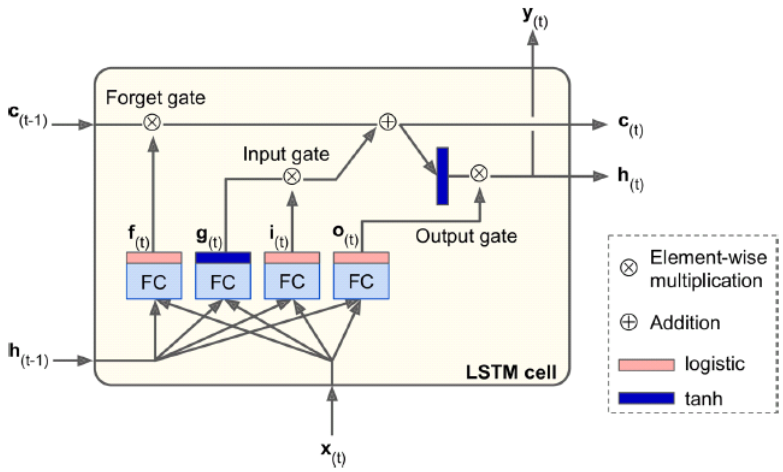
- *LSTM* (*engl. Long Short-Term Memory*) ili duga kratkoročna memorija koja će biti detaljnije objašnjena u sljedećem poglavlju;
- *Elman neuronska mreža* je arhitektura primarno konstruirana za algoritme obrade jezika, no može biti korisna za probleme sekvencijalnih ulaznih podataka;
- *Neural History Compressor* je model koji je djelomično riješio problem nestajanja gradijenata implementacijom kao nenadgledani stog RNN-a tako da ulazni sloj uči predviđati svoj sljedeći unos iz prethodnih unosa.
- *GRU* (*engl. Gated Recurrent Unit*) ili povratna propusna ćelija je pojednostavljena verzija LSTM modela u kojoj imamo samo jedna vrata za kontrolu informacija koje se zaboravljaju i unose;
- *BRNN* (*engl. Bidirectional Recurrent Neural Networks*) – dvosmjerna RNN koja se sastoji od dvije zasebne RNN, posebna za prolaz unaprijed i posebna za natrag, no spojene su na zajednički izlazni sloj. Na taj način, izlazni sloj ima pregled za kompletni prošli i budući kontekst ulazne sekvence;

- *DRNN* (*engl. Deep Recurrent Neural Networks*);*MDRNN* (*engl. Multidimensional Recurrent Neural Networks*).

5.2 LSTM

LSTM (*engl. Long Short-Term Memory*) ili duga kratkoročna memorija je vrsta RNN mreže koja se najčešće koristi za predviđanje vremenskih serija podataka. Kao i RNN, LSTM mreže također imaju povratne veze čime se izlaz prethodnog vremenskog koraka koristi kao kontekst za trenutni ulaz. No, za razliku od ostalih RNN-ova, LSTM ima jedinstvenu formulaciju koja mu omogućuju da izbjegne probleme na koje nailaze ostale RNN mreže. Problem ostalih RNN-a je u ažuriranju težina pri čemu brzo postanu tako male, postanu 0 tako da više nemaju učinka, tzv. nestajući gradijent (*engl. vanishing gradient*) ili postanu velike što rezultira vrlo velikim promjenama, tzv. eksplodirajući gradijent (*engl. exploding gradient*).

LSTM je taj problem riješio dizajnom. Osnovna jedinica LSTM mreže je tzv. memorijska ćelija, memorijski blok ili kraće, ćelija. Na sljedećoj slici je prikazana arhitektura LSTM ćelije.



Slika 5: LSTM ćelija [19]

Ključna ideja mreže je naučiti što spremiti u dugoročnu memoriju, što izbaciti i što iščitati iz spremljenog. Stanje ćelije je osnovni dio LSTM mreže čija je uloga djelovati kao dugotrajna memorija za čuvanje informacija tijekom svih iteracija.

Kao što je vidljivo sa slike LSTM ćelija ima tri ulaza kojima zaprima nove informacije:

- vektor c (*engl. cell state*) – predstavlja stanje ćelije iz prethodne iteracije, dugoročno stanje

- vektor h (*engl. hidden*) – kratkoročno stanje iz prethodnih iteracija u skrivenom sloju
- x – ulazna vrijednost u trenutnoj iteraciji

Sa slike je vidljivo da LSTM ćelija ima vrata. Vrata su zapravo funkcije koje upravljaju protokom informacija u ćeliji. Postoje tri vrste:

- Vrata zaborava – odlučuju koje informacije odbaciti iz ćelije
- Ulazna vrata – odlučuju koje vrijednosti iz ulaza odabrati za ažuriranje memorije
- Izlazna vrata – odlučuju o izlazu temeljem ulaza i memorije ćelije

Vrata zaborava i ulazna rade na ažuriranju unutarnjeg stanja ćelije.

Princip rada: Kako stanje memorije c_{t-1} prolazi kroz mrežu slijeva u desno, prvo prolazi kroz vrata zaborava, izbacuje neke informacije i zatim dodaje nove putem operacije dodavanja u ulaznim vratima. Prilikom svakog koraka, neke informacije se izbacuju, neke dodaju. Po dodavanju, dugoročno stanje se kopira i prosljeđuje tanges funkciji nakon čega se rezultat filtrira u izlaznim vratima. Kao rezultat nastaje kratkoročno stanje h_t koji odgovara izlazu za trenutni vremenski korak ćelije y_t. Ulaz trenutnog vremenskog koraka x_t i prethodno kratkoročno stanje h_t se prosljeđuju u svaki od 4 povezana sloja. Glavni sloj g_t analizira trenutni ulaz x_t i prethodno stanje h_{t-1}. Izlaz sloja se sprema u dugoročno stanje c_t. Ostala tri sloja su kontrolori vrata s logističkim aktivacijskim funkcijama. Ako je izlaz 0, vrata se zatvaraju, ako je izlaz 1, vrata se otvaraju i vrata odrađuju zadanu funkciju, brišu, dodaju i čitaju informacije.

Ukratko, LSTM ćelije mogu naučiti prepoznati bitnu ulaznu informaciju, spremiti je u dugoročnu memoriju, sačuvati na duže vrijeme i izvući kada je potrebno.

Na slici je prikazana samo jedna ćelija za lakše shvaćanje arhitekture i rada, no LSTM sloj se sastoji od skupa povratno povezanih blokova koji se sastoje od ćelija. Mreža može komunicirati s ćelijama samo kroz „LSTM“ vrata.

Tri ključne prednosti LSTM :

- Uspješno prevladavanje tehničkih problema RNN-a, točnije problema padajućeg i eksplodirajućeg gradijenta,
- Posjeduje memoriju kojom premošćuje problem dugoročne vremenske ovisnosti sekvencijalnih podataka,
- Obrađivanje ulaznih i izlaznih sekvencijalnih podataka korak po korak čime su omogućeni ulazni i izlaznih podaci različitih duljina [17].

5.3 Arhitektura LSTM

Standardna arhitektura LSTM mreže, tzv., *Vanilla LSTM*, definirana 1997. sastoji se od ulaznog sloja, LSTM skrivenog sloja i izlaznog sloja.

U slučaju barem 2 skrivena LSTM sloja, pričamo o tzv. *Stacked LSTM*. Dodavanje dodatnog sloja doprinosi rasterećenju mreže time što je potrebno manje neurona te je ubrzano učenje. U tom svjetlu to se može gledati kao optimizacija Vanilla LSTM mreže. Također, time se postiže i stabilnija platforma za zahtjevnije probleme predviđanja vremenskih serija.

CNN (*engl. Convolutional Neural Networks*) je neuronska mreža s konvolucijskim i slojem sažimanja koji olakšavaju obradu podataka u 3 dimenzije: širinu, visinu i dubinu zbog čega se najčešće koriste za obradu slika i računalni vid [16]. *CNN LSTM* se sastoji od CNN sloja za izdvajanje obilježja iz ulaznih podataka te LSTM sloja kao potporu predviđanja sekvenci. Ovaj tip arhitekture je nastao kao odgovor na probleme vizualnih vremenskih serija te se najčešće koristi za generiranje tekstualnog opisa iz niza slika ili automatsko generiranje opisa slike.

Encoder-Decoder LSTM je mreža koja se sastoji dva LSTM modela: jedan za čitanje ulaznih sekvenci i kodiranje u vektor fiksne duljine i drugi dio koji dekodira taj vektor u izlaznu sekvencu. Mreža je razvijena za obradu prirodnog jezika, strojno prevođenje jezika.

Kao i ostale neuronske mreže, LSTM ima široku primjenu, neki od primjera su:

- Automatsko generiranje opisa slike – sustav generira opis sadržaja na danoj slici. Proces kreće od detektiranja objekata na fotografiji i generiranje labela ili oznaka objekata, npr. dijete, lopta, igralište te je sljedeći korak stavljanje tih labela u koherentan opis. Za detekciju objekata na fotografiji se koriste CNN, konvolucijske neuronske mreže i potom se koristi LSTM za postavljanje labela u smislenu rečenicu,
- Automatsko generiranje rukopisa – na temelju skupa primjera rukopisa potrebno generirati novi rukopis za danu riječ ili rečenicu. Rukopis se definira kao niz koordinata kreiran pomoću olovke nastalih prilikom pisanja. Od danog skupa primjera rukopisa uči se odnos između kretnji olovke i slova i time se mogu generirati novi primjeri rukopisa.

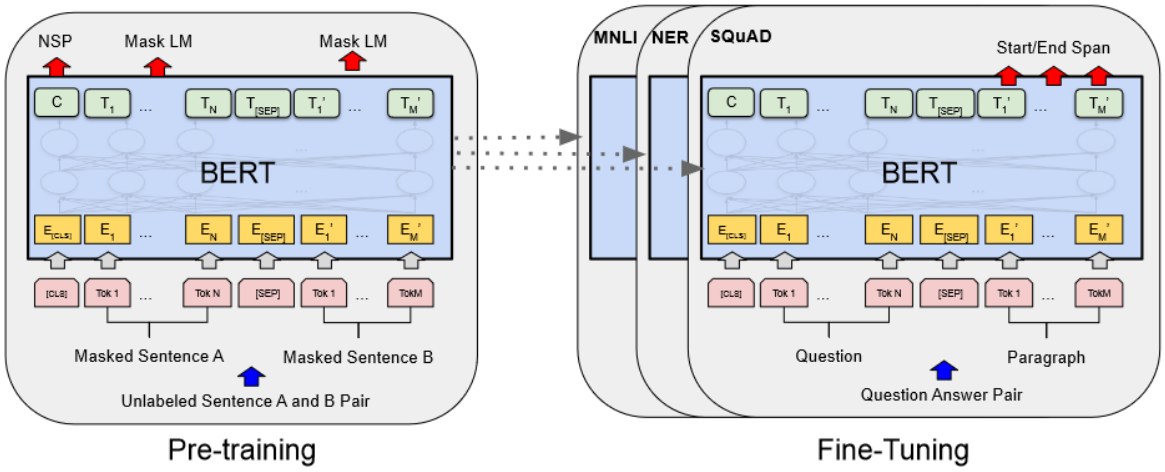
6 Model BERT

BERT *(engl. Bidirectional Encoder Repreentations from Transformers)* je unaprijed naučeni, trenirani model za obradu prirodnog jezika kreiran 2018. u Google. Radi se o novijem algoritmu dubokog učenja na području NLP-a [20].

Jedna od prednosti modela jest njegovo prethodno treniranje na većem setu podataka poput Wikipedije čime se u analizi manjeg skupa podataka postižu točniji rezultati. Druga prednost pred ostalim modelima za obradu jezika dolazi iz arhitekture modela te mu daje mogućnosti čitanja u oba smjera, s lijeva u desno i s desna u lijevo. BERT je prvi dvosmjerni sistem za predtreniranje NLP-a.

Model se temelji na arhitekturi transformera. Transformer je model za duboko učenje također osmišljen u Google za rad sa sekvencijalnim podacima, primjerice strojno prevođenje teksta. Sastoji se od dva dijela, kodera i dekodera koji rade na temelju mehanizma pozornosti, tzv. *self-attention* i unaprijedne neuronske mreže. Transformeri nemaju pojam o vremenu, podaci se ne obrađuju u redoslijedu kao u RNN, već se radi o paralelnoj obradi svih podataka odjednom sa mehanizmom pozornosti, pažnje. Radi se o metodi dohvaćanja sekvencijalne prirode podataka uzimajući u obzir kontekst podataka. Mehanizmom pažnje se može opisati mapiranje različitih pozicija jedne sekvence za izračun prikaza sekvenci [21].

BERT ima dva bitna koraka: prethodno učenje *(engl. pre-training)* i fino prilagođavanje parametara *(engl. fine-tuning)* prikazanih na slici ispod.



Slika 6. Princip rada modela BERT [20]

Kako bi model mogao obavljati širok raspon zadataka, potrebno je ulazne podatke prezentirati sa tokenima. Jedna sekvenca može predstavljati jednu ili više rečenica ili skup nabacanih riječi. Ulazne vrijednosti za model su tokeni. Prvi token svake sekvence je klasifikacijski token [CLS]. Posljednje skriveno stanje koje odgovara tom tokenu se koristi kao agregirana sekvenca za zadatke klasifikacije. Parovi rečenica se unutar sekvence razdvajaju pomoću tokena separatora [SEP] .

U obje procedure, predtreniranju i podešavanju koristi se ista arhitektura. Isti parametri kao što su u unaprijednom učenju, koriste se i prilikom prilagođavanja modela za rad na jednostavnijim primjerima. Tijekom faze prethodnog učenja, model se uči na neoznačenim podacima kroz dvije nenadzirane tehnike, tzv. maskirani LM, MLM *(engl. masked LM)* i predikciju sljedeće rečenice, tzv. NSP *(engl. next sentence prediction)*. Princip je sljedeći: 15% ulaznih podataka se maskira oznakom [MASK] i potom se pokušavaju predvidjeti maskirani podaci. Taj postupak se naziva maskirani ML gdje algoritam gleda maskirani podatak i njegove lijeve i desne susjedne podatke. Sljedeća tehnika se bavi utvrđivanjem odnosa između rečenice A i rečenice B kako bi se razumio kontekst slijeda B iza A [20].

Postoje 4 unaprijed naučene verzije modela te u sklopu rada koristimo model bert-base-uncased.

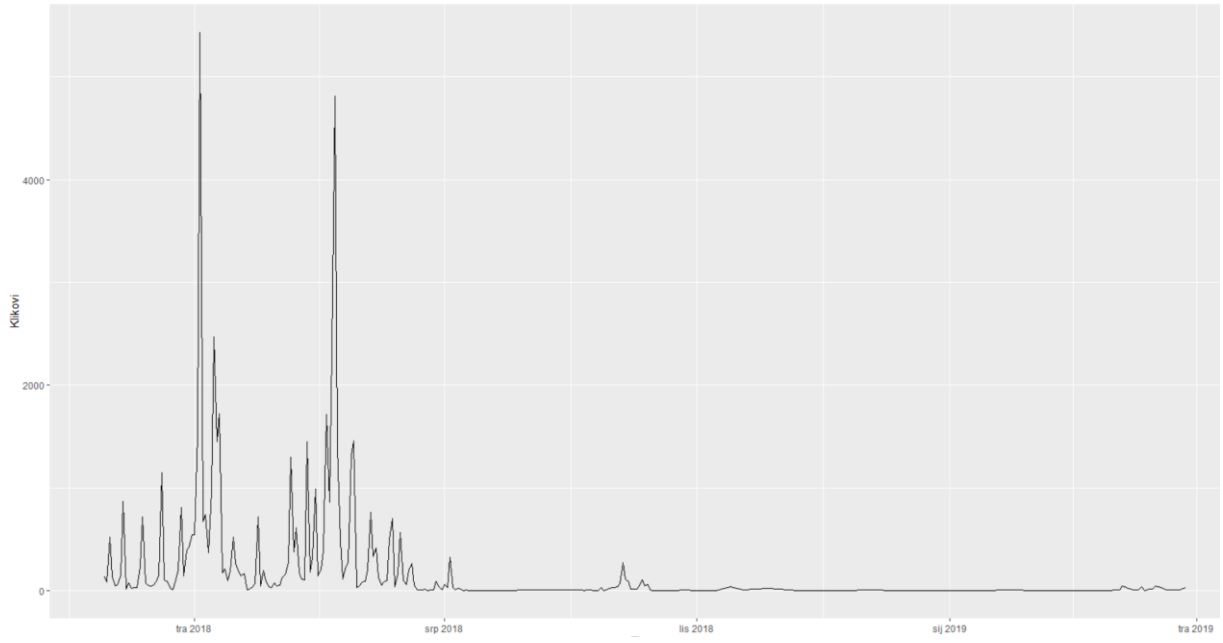
BERT model se koristi kod automatskog odgovaranja na pitanja (*Q&A*), analize sentimenta, klasifikacije teksta, koristi se u tražilici Googlea, a Facebook je razvio svoju verziju modela, tzv. RoBERTa za klasifikaciju i filtriranje postova [22].

7 Projektni zadatak

7.1 Skup podataka

Podaci korišteni za analizu su logovi (zapisi aktivnosti) iz LMS-a Sveučilišta u Rijeci. Obrađeni su podaci sa predmeta Programiranje 2 tijekom jednog semestra 2017./2018. akademske godine.

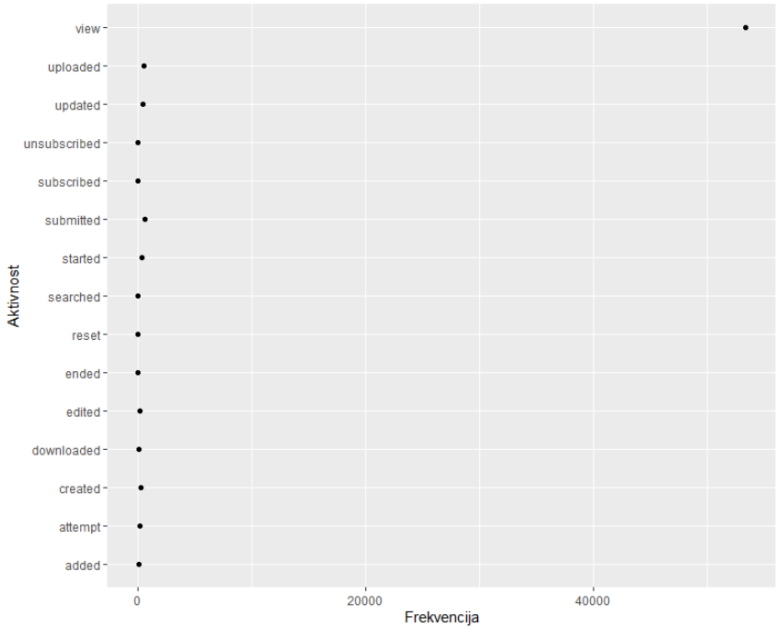
Kako se radi o standardnim logovima, set podataka sadrži sljedeće atribute: Vrijeme, Puno ime, Odnosi se na, Aktivnost na, Komponenta, Naziv aktivnosti, Opis, Izvor, IP adresa. Inicijalni skup podataka sadrži 68976 zapisa. Izvadak logova iz sustava sadrži sve logove od početka kreiranja e-kolegija od strane akademske administracije i IT podrške preko zapisa aktivnosti koje su odradili studenti do kreiranja i ocjenjivanje testova od strane profesora. Stoga je potrebno odraditi pripremu podataka u kojoj su osjetljivi podaci anonimizirani i izbačeni te zadržani samo logovi studenta. Time se broj zapisa smanjio na 56017 te brojimo 105 studenata koji su u uzorku. Radi se o manjem uzorku ako uzimamo u obzir broj studenata, što bi iznosilo oko 500 zapisa po studentu.



Slika 7: Grafički prikaz aktivnosti na kolegiju kroz vrijeme

U pripremi podataka iz polja opis je izvučen generalni naziv aktivnosti poput *view* (pregled sadržaja e-kolegija, izvještaja, zapisa uživo), *submitted* (predaja kolokvija, zadaća i testova),

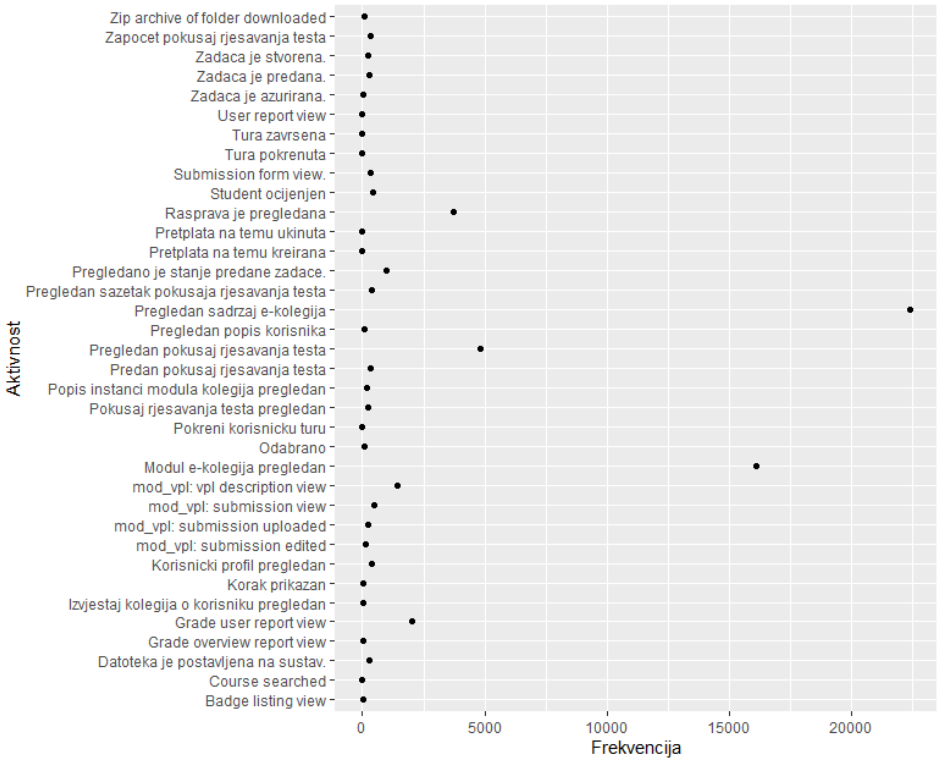
added (davanje suglasnosti s ocjenom, prijava za nadoknadu/ispravak). Od generalnih aktivnosti, najviše zapisa ima aktivnost *view, tj*. 95% logova aktivnosti studenata je *view*.



	Aktivnost	Frekvencija
1	added	101
2	attempt	206
3	created	249
4	downloaded	60
5	edited	121
6	ended	2
7	reset	2
8	searched	1
9	started	337
10	submitted	618
11	subscribed	1
12	unsubscribed	1
13	updated	458
14	uploaded	493
15	view	53367

Slika 8: Prikaz aktivnosti studenata

Unutar aktivnosti *view* najviše zapisa ima *Naziv aktivnosti: Pregledan sadrzaj e-kolegija* – 22375 kao što je vidljivo na slici ispod. Kao što primjećujemo neke aktivnosti su zastupljenije od drugih što će imati utjecaj i na točnost modela.



Slika 9: Prikaz frekvencije Naziva aktivnosti unutar view

7.2 Analiza

7.2.1 Predviđanje aktivnosti pomoću LSTM arhitekture

U ovom poglavlju je prikazano modeliranje studenata i predikcija u LSTM [23]. Ulazni podaci su logovi studenata u datoteci *Data1.csv*. najprije se učitavaju potrebne biblioteke te csv datoteke sa separatorom. Dobra je praksa provjeriti prvih 5 zapisa učitane datoteke. Za unos podataka koristi se biblioteka *pandas*.

```
#importing the libraries
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import io
%matplotlib inline
```

```
[2] from google.colab import drive
    drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
[3] btc = pd.read_csv("/content/gdrive/My Drive/Data1.csv",error_bad_lines=False, sep= " ")
```

```
btc.head()
```

	Time	StudentAno	ActivityOn	Komponenta	NameActivity	Activity
1	2019-03-28	Student060	Mapa: Priprema za kviz - razni primjeri	Mapa	Modul e-kolegija pregledan	view
2	2019-03-28	Student060	Datoteka: Priprema za kviz	Datoteka	Modul e-kolegija pregledan	view
3	2019-03-28	Student060	E-kolegij: Programiranje 2	Sustav	Pregledan sadrzaj e-kolegija	view
4	2019-03-28	Student060	Datoteka: Priprema za kviz - Uvod u pokazivace 2	Datoteka	Modul e-kolegija pregledan	view
5	2019-03-28	Student060	Datoteka: Priprema za kviz - Uvod u pokazivace 1	Datoteka	Modul e-kolegija pregledan	view

Slika 10. Učitavanje biblioteka i podataka

Za analizu aktivnosti odabire se atribut *ActivityOn* jer je najviše heterogen, ima 130 jedinstvenih vrijednosti. Ti podaci su kategorički te ih je potrebno transformirati u numeričke vrijednosti za što se koristi biblioteka *sklearn*.

```
[14] from sklearn.preprocessing import LabelEncoder
     lbl=LabelEncoder()
```

```
btc['ActivityOn']=lbl.fit_transform(btc['ActivityOn']) #tranform to numeric
btc['ActivityOn'].unique()
```

```
array([ 92,  44,  86,  48,  47,  42,  54,  78,  45,  59,  22,  38,  37,
         4, 113,  70,  62,  13,  36, 106, 105,  97,  31,  21, 111,  73,
        77,  71,  34,  88,  52,  12,  51,  35,  56,  94,  87, 112,  29,
        53,  89,  28,  27,  10,  41,  30,   9,  43,  26,  25,  24,  23,
        33,  32, 116, 107,  75,  91,  76,  95,  90,   8,  60, 129,  50,
        16,  17,   6,  58,  96,   0,   1,   5,   7,   3,   2,  46,  11,
        82, 118, 110,  63, 117, 109, 108,  49,  57,  40,  39,  79,  80,
        81,  83,  61,  93, 104, 115, 114,  64,  68,  84,  72,  19, 128,
       127, 126, 125,  66,  65,  74,  15,  14,  20,  67,  69,  55,  18,
       124, 123, 122, 121, 120, 119, 102, 100,  99,  98, 103, 101,  85])
```

Slika 11. Transformacija kategoričkih atributa naziva aktivnosti

Provodi se skaliranje za ograničenje podataka u opsegu (1-,1) te se učitani podaci dijele u set za učenje i testiranje.

```
#we now split the data into training and test set
import numpy as np
#train set
X_train  = np.array(X[:1000])
y_train = np.array(y[:1000])

#test set
X_test = np.array(X[1000:])
y_test = np.array(y[1000:])

print("X_train size: {}".format(X_train.shape))
print("y_train size: {}".format(y_train.shape))
print("X_test size: {}".format(X_test.shape))
print("y_test size: {}".format(y_test.shape))
```

```
X_train size: (1000, 7, 1)
y_train size: (1000, 1)
X_test size: (55010, 7, 1)
y_test size: (55010, 1)
```

Slika 12. Podjela podataka za učenje i testiranje

Određuju se parametri mreže *(batch_size, window_size, hidden_layer i learning rate),* implementacija težina ulaznih vrata, vrata zaborava, izlaza, ćelije, te se definira LSTM ćelija koja vraća stanje ćelije i skriveno stanje kao izlazna vrijednost te petlja za svaki prolaz kroz mrežu.

```
#Weights for the input gate
weights_input_gate = tf.Variable(tf.truncated_normal([1, hidden_layer], stddev=0.05))
weights_input_hidden = tf.Variable(tf.truncated_normal([hidden_layer, hidden_layer], stddev=0.05))
bias_input = tf.Variable(tf.zeros([hidden_layer]))
```

Slika 13. Definiranje težina za vrata LSTM ćelije

```
def LSTM_cell(input, output, state):

    input_gate = tf.sigmoid(tf.matmul(input, weights_input_gate) + tf.matmul(output, weights_input_hidden) + bias_input)

    forget_gate = tf.sigmoid(tf.matmul(input, weights_forget_gate) + tf.matmul(output, weights_forget_hidden) + bias_forget)

    output_gate = tf.sigmoid(tf.matmul(input, weights_output_gate) + tf.matmul(output, weights_output_hidden) + bias_output)

    memory_cell = tf.tanh(tf.matmul(input, weights_memory_cell) + tf.matmul(output, weights_memory_cell_hidden) + bias_memory_cell)

    state = state * forget_gate + input_gate * memory_cell

    output = output_gate * tf.tanh(state)
    return state, output
```

Slika 14. Definiranje LSTM ćelije

Nakon predviđanja izlazne vrijednosti, definira se funkcija gubitka *(engl. loss function)* – srednja kvadratna greška *(engl. mean squared error, MSE).* Kako bi se izbjegao problem eksplodirajućeg gradijenta radi se tzv. gradient clipping – definiranjem vrijednosti prema kojoj gradijent teži. Za optimizaciju se koristi algoritam Adam.

Početno je definirano 40000 zapisa u set za treniranje (što odgovara podjeli 30% podataka za test i 70% za učenje) u 200 epoha. Takvo učenje se nije moglo izvršiti u Google Colab te je i nakon pola sata učenje mreže još uvijek radilo na drugoj epohi. Posljedično, odabrano je 1000 uzoraka za učenje te je greška predikcije takvog modela 20% što je dobar rezultat (Slika 15).

```
#we now train the network
session = tf.Session()
session.run(tf.global_variables_initializer())
for i in range(epochs):
    traind_scores = []
    ii = 0
    epoch_loss = []
    while(ii + batch_size) <= len(X_train):
        X_batch = X_train[ii:ii+batch_size]
        y_batch = y_train[ii:ii+batch_size]

        o, c, _ = session.run([outputs, loss, trained_optimizer], feed_dict={inputs:X_batch, targets:y_batch})

        epoch_loss.append(c)
        traind_scores.append(o)
        ii += batch_size
    if (i % 30) == 0:
        print('Epoch {}/{}'.format(i, epochs), ' Current loss: {}'.format(np.mean(epoch_loss)))
```

```
Epoch 0/200  Current loss: 0.7685125470161438
Epoch 30/200  Current loss: 0.6278237104415894
Epoch 60/200  Current loss: 0.435150146484375
Epoch 90/200  Current loss: 0.29663389921188354
Epoch 120/200  Current loss: 0.2375897616147995
Epoch 150/200  Current loss: 0.24541226029396057
Epoch 180/200  Current loss: 0.20265009999275208
```

Slika 15. Greška modela

7.2.2 Predviđanje uspjeha pomoću BERT modela

BERT model ćemo iskoristiti za klasifikaciju uspjeha na kolegiju. Izvorni kod je preuzet sa znanstvenog web sjedišta „*Towards data science*“ [24]. U tu svrhu potrebna je jedna sekvenca koja će biti klasificirana. U zasebnoj skripti za pripremu podataka, kombinirajući logove studenata i njihove uspjehe, datoteke *Data1.csv* i *Grades.csv* kreiramo novu csv datoteku sa 105 zapisa. Svaki zapis se odnosi na jednog studenta, sve njegove aktivnosti iz dvije kolone *ActivityOn* i *Activity* te konačni uspjeh (*Pass* ili *Fail*).

Za korištenje BERT modela potrebno je instalirati paket *transformers* i ostale potrebne biblioteke. Koristit će se algoritam BertForSequenceClassification za klasifikaciju nizova aktivnosti.

```
[8]  ! pip install transformers
```

```
[11] import torch
     from tqdm.notebook import tqdm

     from transformers import BertTokenizer
     from torch.utils.data import TensorDataset

     from transformers import BertForSequenceClassification

     df.head()
```

	ActivityOn_Activity	result
0	Forum: Forum s vijestima view Forum: Forum s v...	Pass
1	Forum: Forum s vijestima view Odabir: Suglasno...	Pass
2	E-kolegij: Programiranje 2 view E-kolegij: Pro...	Pass
3	Forum: Forum s vijestima view Forum: Forum s v...	Fail
4	Datoteka: Ogledni primjer online provjere - Kr...	Pass

Slika 16. Instalacija paketa i prikaz skupa podataka

Pass i *Fail* labele se transformiraju u numeričke vrijednosti, sa *Pass:0* i *Fail:1*.

```
possible_labels = df.result.unique()

label_dict = {}
for index, possible_label in enumerate(possible_labels):
    label_dict[possible_label] = index
label_dict
```

```
{'Fail': 1, 'Pass': 0}
```

Slika 17. Atributi Fail/Pass

Potom se skup podataka dijeli na setove *train* za učenje i *val* za testiranje. Od 105 zapisa, 66 ima atribut *Pass*, te 39 *Fail*. Za nizove aktivnosti koji su rezultirali sa neuspjehom – *Fail*, imamo 33 zapisa za učenje i 6 zapisa za testiranje te za nizove aktivnosti koji su rezultirali uspjehom – *Pass*, 56 zapisa je za učenje i 10 zapisa za testiranje.

			ActivityOn_Activity
result	label	data_type	
Fail	1	train	33
		val	6
Pass	0	train	56
		val	10

Slika 18. Skupovi za učenje i testiranje

Koristi se BertTokenizer za kodiranje podataka, stavljanja tokena u sekvence za treniranje i testiranje te se kreira set podataka *datatest_train* i *dataset_val* koji će se proslijediti predtreniranom modelu.

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased',
                                          do_lower_case=True)

encoded_data_train = tokenizer.batch_encode_plus(
    df[df.data_type=='train'].ActivityOn_Activity.values,
    add_special_tokens=True,
    return_attention_mask=True,
    pad_to_max_length=True,
    max_length=256,
    return_tensors='pt'
)

encoded_data_val = tokenizer.batch_encode_plus(
    df[df.data_type=='val'].ActivityOn_Activity.values,
    add_special_tokens=True,
    return_attention_mask=True,
    pad_to_max_length=True,
    max_length=256,
    return_tensors='pt'
)


input_ids_train = encoded_data_train['input_ids']
attention_masks_train = encoded_data_train['attention_mask']
labels_train = torch.tensor(df[df.data_type=='train'].label.values)

input_ids_val = encoded_data_val['input_ids']
attention_masks_val = encoded_data_val['attention_mask']
labels_val = torch.tensor(df[df.data_type=='val'].label.values)

dataset_train = TensorDataset(input_ids_train, attention_masks_train, labels_train)
dataset_val = TensorDataset(input_ids_val, attention_masks_val, labels_val)
```

```
input_ids_train
```

```
tensor([[  101,  7057,  1024,  ..., 17706,  1024,   102],
        [  101,  7057,  1024,  ..., 12184,  2912,   102],
        [  101,  1041,  1011,  ..., 23233,  6559,   102],
        ...,
        [  101,  1041,  1011,  ...,  6460, 16643,   102],
        [  101,  1041,  1011,  ...,  1055,  6819,   102],
        [  101,  7057,  1024,  ...,  5737,  3501,   102]])
```

Slika 19. Dodavanje tokena

Kao predtrenirani BERT model, koristi se osnovni model tzv. *bert-base-uncased*. Podaci se učitavaju u *dataloader_train* i *dataloader_validation* te se definiraju optimizacija Adam, učenje u 5 epoha. Iz biblioteke sklearn.metrics se uvodi f1_score_func kao mjera točnosti modela. Model se učitava na uređaj te vidimo sve njegove slojeve, *BertAttention, BertIntermediate, BertOutput.*

```
model.to(device)
```

```
          )
          (1): BertLayer(
            (attention): BertAttention(
              (self): BertSelfAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
              (output): BertSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
            )
            (intermediate): BertIntermediate(
              (dense): Linear(in_features=768, out_features=3072, bias=True)
            )
            (output): BertOutput(
              (dense): Linear(in_features=3072, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (2): BertLayer(
            (attention): BertAttention(
              (self): BertSelfAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
              (output): BertSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
            )
            (intermediate): BertIntermediate(
```

Slika 20. Predtrenirani model

Slijedi evaluacija podataka i učenje modela. Po završetku se bira model sa najvećim *F1 Score* epohe na kojem radimo klasifikaciju. Za epohe 3, 4 i 5 je dobiveni rezultati *F1 Score* isti: 0.875, što znači da je točnost modela 87,5%. Od navedena tri biramo model epohe 5 jer ujedno sadrži najmanje greške modela za skupove treniranja: 38% i validacije: 33%.

```
100% 5/5 [20:29<00:00, 253.02s/it]

Epoch 1
Training loss: 0.6659024208784103
Validation loss: 0.6390304168065389
F1 Score (Weighted): 0.4807692307692308


Epoch 2
Training loss: 0.6140544990698497
Validation loss: 0.507727175951004
F1 Score (Weighted): 0.7934782608695653


Epoch 3
Training loss: 0.5716753035783768
Validation loss: 0.3384079684813817
F1 Score (Weighted): 0.875


Epoch 4
Training loss: 0.4540011465549469
Validation loss: 0.32800056288639706
F1 Score (Weighted): 0.875


Epoch 5
Training loss: 0.3886374940474828
Validation loss: 0.3373254984617233
F1 Score (Weighted): 0.875
```

Slika 21. Rezultat učenja BERT modela

Model epohe 5 se koristi za algoritam BertForSequenceClassification za klasifikaciju *Pass/Fail* na skupu podataka za testiranje te postiže sljedeće rezultate: za klasu *Pass*, točnost modela je 9 od 10 – 90% te za klasu *Fail* je točnost 5 od 6 – 83% što su dobri rezultati.

```
model = BertForSequenceClassification.from_pretrained("bert-base-uncased",
                                                      num_labels=len(label_dict),
                                                      output_attentions=False,
                                                      output_hidden_states=False)

model.to(device)

model.load_state_dict(torch.load('finetuned_BERT_epoch_5.model', map_location=torch.device('cpu')))

_, predictions, true_vals = evaluate(dataloader_validation)
accuracy_per_class(predictions, true_vals)
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequence
seq_relationship.bias', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.bias', 'cls.predicti
'cls.predictions.transform.dense.bias', 'cls.predictions.transform.dense.weight', 'cls.seq_relationship.w
ns.transform.LayerNorm.weight']
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model t
or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTr
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a mod
be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassifica
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base
initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and infer

Class: Pass
Accuracy: 9/10

Class: Fail
Accuracy: 5/6
```

Slika 22. Rezultati klasifikacije Pass/Fail

7.2.3 Predviđanje uspjeha pomoću LSTM arhitekture

U ovom primjeru se uspjeh studenata predviđa na temelju bodova koje ostvare tijekom semestra i broja jedinstvenih atributa po studentu. Uvode se sve potrebne biblioteke te se radi na kreiranju jednog podatkovnog okvira sa dvije csv datoteke, *Data1.csv* i *Grades.csv*.

Iz logova se analiziraju jedinstveni zapisi svakog studenta po pojedinom atributu, npr. Student060 je bio aktivan 44 dana (zabilježen u sustavu kroz 44 različita datuma, atribut *Time*), broji 79 jedinstvenih zapisa za *ActivityOn*, 13 jedinstvenih zapisa po atributu *Komponenta* te 23 jedinstvena zapisa po *NameActivity*. Za svakog studenta se kreira zapis njegovih jedinstvenih (*engl. distinct*) vrijednosti.

Podatkovni okvir broja jedinstvenih atributa po studentu se spaja sa bodovima skupljenim kroz trajanje kolegija te se izbacuju redundantni podaci. Ocjena, *Grade* (A do F) je transformirana u numeričku vrijednost te se po *Fail/Pass* napravi konkatenacija čime se izjednači broj *Fail/Pass* varijabli.

Kreira se matrica korelacije za bolji prikaz odnosa između varijabli te se izbacuje atribut *Grades* zbog visoke korelacije sa ostalim atributima.

Slika 23. Matrica korelacije

Za standardizaciju podataka se koristi *MinMaxScaler* te kreira set za učenje i set za testiranje.

Kreira se jednostavni sekvencijalni model korištenjem klase Sequential i jednostavni LSTM model sa jednim LSTM slojem od 25 neurona i sigmoidalne aktivacijske funkcije te dva Dense sloja, jedan sa 10 neurona i jedan sa 1 neuronom. Definiramo model sa loss funkcijom *binary_crossentropy* koja se koristi u zadacima klasifikacije, optimizacijsku funkciju *sgd* i kao mjeru točnosti modela *accuracy.* Model se trenira u batchevima od 10 uzorka u 500 epoha.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, LSTM, GRU
from tensorflow.keras.callbacks import EarlyStopping

model = Sequential()
model.add(LSTM(25,return_sequences=True,activation='sigmoid',input_shape=(X_test.shape[1],1)))
model.add(Dense(units=10,activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))


model.compile(loss='binary_crossentropy',optimizer='sgd',metrics=['accuracy'])
model.fit(X_train,y_train,batch_size=10,epochs=500)
```

Slika 24. Kreiranje modela za učenje

Model ima preciznost od 77,40% što je dobar rezultat. Može se zaključiti da postoji poveznica između broja jedinstvenih aktivnosti provedenih na LMS i uspjeha na kolegiju.

```
Epoch 495/500
11/11 [==============================] - 0s 7ms/step - loss: 0.5247 - accuracy: 0.7723
Epoch 496/500
11/11 [==============================] - 0s 7ms/step - loss: 0.5233 - accuracy: 0.7905
Epoch 497/500
11/11 [==============================] - 0s 8ms/step - loss: 0.5214 - accuracy: 0.7870
Epoch 498/500
11/11 [==============================] - 0s 7ms/step - loss: 0.5212 - accuracy: 0.7766
Epoch 499/500
11/11 [==============================] - 0s 7ms/step - loss: 0.5190 - accuracy: 0.7732
Epoch 500/500
11/11 [==============================] - 0s 7ms/step - loss: 0.5187 - accuracy: 0.7740
<keras.callbacks.History at 0x7f21541ebe10>
```

```
[70] model.summary()

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 11, 25)            2700
_________________________________________________________________
dense_2 (Dense)              (None, 11, 10)            260
_________________________________________________________________
dense_3 (Dense)              (None, 11, 1)             11
=================================================================
Total params: 2,971
Trainable params: 2,971
Non-trainable params: 0
```

Slika 25. Rezultati modela

7.3 Korištene tehnologije

Od tehnologija su korišteni Rstudio unutar otvorenog softvera Anaconde na operativnom sustavu Windows te Google Colaboratory.

Anaconda je platforma namijenjena podatkovnim znanstvenicima. Radi se najpopularnijoj distribuciji programskih jezika Python i R za analizu podataka i strojno učenje. Trenutno broji 25 milijuna korisnika u preko 235 zemalja [25].

U pripremi i vizualizaciji podataka je korišten programski jezik R u razvojnom okruženju Rstudio unutar Anaconde. Programski jezik R se često koristi u svrhu statističkih izračuna, analize i vizualizacije podataka. Pruža velik skup statističkih i grafičkih tehnika te se može proširiti raznim paketima [26].

Jupyter notebook jest još jedna od aplikacija koja je dostupna u softveru Anaconde, no u ovom radu je korištena na proizvodu Google Colabratory, kraće Colab. Jupyter notebook je razvojno okruženje za programski jezik Python. Python je programski jezik koji se također koristi u razvoju programa u strojnom učenju i analizi podataka. Google Colab je Jupyter notebook spremljen na Google Drive-u. Sastoji se od ćelija koje omogućuju jasniji pregled koda te je u dokumentu moguće pokretati Python bez nekih potrebnih konfiguracija pošto se sam kod pokreće i izvršava na Google cloud serverima što omogućava bržu i „jeftiniju“ (manju potrošnju memorije) obradu. Google Colab ima integrirane biblioteke za analizu podataka, za izradu rada je potrebno koristiti biblioteku Keras [27]. Biblioteka Keras je jedna od najčešće korištenih sučelja *(engl. Framework)* za rad sa neuronskim mrežama. Keras je dio krovne okoline TensorFlow, platforme za strojno učenje kreirane u Google-u [28].

8 Zaključak

Ostavljamo digitalne tragove posvuda na internetu, čak i kada toga nismo ni svjesni, i čitajući dnevni portal i pretraživanjem automobila na tražilici ili fotografiranjem u kafiću. Mnogi servisi znaju i umiju iskoristiti te mrvice kako bi ostvarili profit.

Edukacija se isto polako uključuje u to područje sa već spomenutim disciplinama Analitikom učenja i Analizom podataka u obrazovanju. Ono što se iz priloženog istraživanja može zaključiti jest da se iz logova mogu izvući korisne informacije i da vrijeme provedeno na sustavu za e-učenje utječe i na uspjeh studenata. Naravno, na uspjeh i predviđanje sljedeće aktivnosti utječu i mnogi drugi faktori, no generalni zaključak jest da se podaci mogu iskoristiti u dodatnu pomoć prilikom obrazovanja poput automatiziranih preporuka sustava kada je student predugo vremena neaktivan ili predlaganje aktivnosti ili resursa u svrhu ostvarivanja boljih rezultata. Drugim riječima, sustav preporuke ima za cilj bolji uspjeh. Što se tiče tehnologija i LSTM kojeg je BART model izgurao sa scene u NLP-u daju podjednako dobre rezultate. Napretkom tehnologije i razvijanjem novih algoritama, modeli će napredovati, biti će kompleksniji i moći će obrađivati kompleksnije zadatke te će se time omogućiti i bolja, jasnija obrada podataka i u obrazovanju. Možda i rukopis može biti u korelaciji sa uspjehom u školi?

.

9 Popis literature

[1] Charles Lang, George Siemens, Alyssa Wise, Dragan Gašević, The Handbook of Learning Analytics, SOLAR, 2017.

[2] »What is e-learning,« [Mrežno]. Available: http://www.elearningnc.gov/about_elearning/what_is_elearning/. [Pokušaj pristupa 16 9 2019].

[3] »What Is Distance Education? - Definition & History,« Study.com, 6 August 2020. [Mrežno]. Available: https://study.com/academy/lesson/what-is-distance-education-definition-history.html. [Pokušaj pristupa 08 08 2020].

[4] »Moodle,« [Mrežno]. Available: https://moodle.com. [Pokušaj pristupa 19 07 2021].

[5] »Merlin,« [Mrežno]. Available: https://moodle.srce.hr/2018-2019/. [Pokušaj pristupa 5 10 2019].

[6] »CENTAR ZA E-UČENJE,« 50.srce, [Mrežno]. Available: https://50.srce.hr/ceu.html. [Pokušaj pristupa 16 7 2021].

[7] S. Downes, »Connectivism and Connective Knowledge,« National Research Council, Canada, 2012.

[8] C. Romero, S. Ventura, »Educational data mining and learning analytics: An updated survey,« *WIREs Data Mining Knowl Discov.*, br. e1355, p. 10, 2020.

[9] A. Geron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly, 2017.

[10] Taweh Beysolow ll, Introduction to Deep Learning Using R, Apress, 2017.

[11] Adam Gibson, Josh Patterson, Deep Learning, O'Reilly Media, Inc., 2017.

[12] D. Kriesel, »A Brief Introduction to Neural Networks,« 2007. [Mrežno]. Available: http://www.dkriesel.com. [Pokušaj pristupa 12 10 2019].

[13] [Mrežno]. Available: https://web.math.pmf.unizg.hr/nastava/su/index.php/download_file/-/view/109/. [Pokušaj pristupa 19 10 2019].

[14] A. Graves, Supervised Sequence Labelling with Recurrent Neural Networks, Springer, 2012.

[15] Nitin Kumar Kain, »Medium,« [Mrežno]. Available: https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f. [Pokušaj pristupa 26 10 2019].

[16] Tin Krambeger, Bojan Nožica, Ivica Dodig, Davor Cafuta, »PREGLED TEHNOLOGIJA U NEURONSKIM MREŽAMA,« *Polytechnic and Design,* svez. 7, 2019.

[17] J. Brownlee, Long Short- Term Memory Networks With Python, 2019.

[18] N. Cvijetic, »Nvidia,« 22 05 2019. [Mrežno]. Available: https://blogs.nvidia.com/blog/2019/05/22/drive-labs-predicting-future-motion/. [Pokušaj pristupa 8 27 2020].

[19] L. Cell. [Mrežno]. Available: https://tekworld.org/2018/12/30/day-48-100-days-mlcode-lstm/#page-content. [Pokušaj pristupa 01 09 2020].

[20] Google AI Language: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, »BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,« u *NAACL*, 2019.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, »Attention is all you need,« u *31st International Conference on Neural Information Processing Systems (NIPS'17)*, New York, 2017. .

[22] M. Cohen-Yashar, »Medium,« [Mrežno]. Available: https://medium.com/swlh/nlu-for-everyone-with-bert-7bedaa609a61. [Pokušaj pristupa 17 8 2021].

[23] S. Ravichandiran, Hands-On Deep Learning Algorithms with Python, Birmingham: Packt Publishing Ltd, 2019.

[24] S. Li, »Multi Class Text Classification With Deep Learning Using BERT,« [Mrežno]. Available: https://towardsdatascience.com/multi-class-text-classification-with-deep-learning-using-bert-b59ca2f5c613. [Pokušaj pristupa 7 9 2021].

[25] »Anaconda,« [Mrežno]. Available: https://www.anaconda.com/. [Pokušaj pristupa 15 8 2021].

[26] »R project,« [Mrežno]. Available: https://www.r-project.org/about.html. [Pokušaj pristupa 15 8 2021].

[27] »Welcome to Colabratory,« 15 8 2021. [Mrežno]. Available: https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=GJBs_flRovLc.

[28] »TensorFlow,« [Mrežno]. Available: https://www.tensorflow.org/. [Pokušaj pristupa 20 8 2021].

Popis slika