**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Kristijan Vrabec

Dubinska analiza podataka u svrhu generiranja kviz pitanja

DIPLOMSKI RAD br. I 969

Zagreb, rujan, 2018.

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**STRUČNI STUDIJ RAČUNARSTVA**

Kristijan Vrabec

JMBAG: 0246051246

Dubinska analiza podataka u svrhu generiranja kviz pitanja

DIPLOMSKI RAD br. I 969

Povjerenstvo:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zagreb, rujan, 2018.

A close up of text on a white background

Description generated with high confidence

**Sažetak**

U ovom dokumentu opisana je dubinska analiza podataka i tehnologije u kojima se ona obavlja kao i strojno učenje i pripadajuće tehnologije. Opisan proces izrade praktičnog rada u programskom jeziku Python, označavanja rečenica NER oznakama, generiranja pitanja iz tih rečenica i prikaza na korisničkom sučelju razvijenom u programskom jeziku C#.

Sadržaj

[1 Uvod 7](#_Toc532148279)

[2 Dubinska analiza podataka 8](#_Toc532148280)

[2.1 Proces dubinske analize podataka 8](#_Toc532148281)

[2.1.1 Prikupljanje podataka 9](#_Toc532148283)

[2.1.2 Izdvajanje podataka 10](#_Toc532148284)

[2.1.3 Normalizacija podataka 10](#_Toc532148285)

[2.1.4 Obrada podataka 11](#_Toc532148286)

[2.1.5 Analiza podataka 12](#_Toc532148287)

[2.1.6 Izvlačenje znanja 12](#_Toc532148288)

[2.2 Tehnologije dubinske analize podataka 13](#_Toc532148289)

[2.2.1 Statistika 13](#_Toc532148290)

[2.2.2 Strojno učenje 14](#_Toc532148291)

[2.2.3 Baze i skladišta podataka 14](#_Toc532148292)

[2.2.4 Izvlačenje informacija 14](#_Toc532148293)

[2.3 Primjene dubinske analize podataka 15](#_Toc532148294)

[2.3.1 Dubinska analiza teksta 15](#_Toc532148295)

[2.3.2 Medicina 15](#_Toc532148296)

[2.3.3 Procjena rizika 15](#_Toc532148297)

[2.3.4 Marketing 15](#_Toc532148298)

[3 Proces strojnog učenja 16](#_Toc532148299)

[3.1 Strojno učenje 16](#_Toc532148300)

[3.2 Vrste strojnog učenja 17](#_Toc532148301)

[3.3 Koraci strojnog učenja 19](#_Toc532148302)

[3.3.1 Priprema i prilagodba podataka za strojno učenje 19](#_Toc532148303)

[3.3.2 Kreiranje modela 20](#_Toc532148304)

[3.3.3 Treniranje modela 23](#_Toc532148305)

[3.3.4 Evaluacija modela 24](#_Toc532148306)

[3.3.5 Mijenjanje parametara ovisno o rezultatima 24](#_Toc532148307)

[4 Tehnologije strojnog učenja 26](#_Toc532148308)

[4.1 Lokalne tehnologije strojnog učenja 26](#_Toc532148309)

[4.1.1 TensorFlow 26](#_Toc532148315)

[4.1.2 Keras 28](#_Toc532148318)

[4.1.3 Caffe 29](#_Toc532148319)

[4.1.4 Usporedba biblioteka i programskih okvira 29](#_Toc532148320)

[4.2 Tehnologije strojnog učenja u oblaku 30](#_Toc532148321)

[4.2.1 Amazon Web Services 30](#_Toc532148322)

[4.2.2 Google Cloud 31](#_Toc532148324)

[4.2.3 Azure Machine Learning Studio 31](#_Toc532148325)

[4.2.4 Usporedba tehnologija za strojno učenje u oblaku 31](#_Toc532148326)

[5 Prepoznavanje imenovanih entiteta 32](#_Toc532148327)

[6 Python 33](#_Toc532148328)

[6.1 Instalacija Python-a 34](#_Toc532148329)

[7 Microsoft Visual Studio 36](#_Toc532148330)

[7.1 Instalacija Microsoft Visual Studio-a 36](#_Toc532148331)

[8 Praktični rad 37](#_Toc532148332)

[9 Zaključak 42](#_Toc532148333)

[Literatura 50](#_Toc532148334)

**Kratice i oznake**

NLTK – Alat za obradu prirodnog jezika (eng. Natural Language toolkit)

UCI – Sveučilište Kalifornije - Irvine (eng. University of California - Irvine)

HTML – Hipertekstualni jezik za označavanje (eng. HyperText Markup Language)

GloVe - globalni vektor za reprezentaciju riječi (eng. Global Vectors for Word Representation).

POS – Vrste riječi (eng. Part-of-speech)

IR – Izvlačenje informacija (eng. Information retrieval)

SQL – Strukturirani jezik za upite (eng. Structured Query Language)

AI – Umjetna inteligencija (eng. Artificial intelligence)

CNTK – Alat za računske mreže (eng. Computational Network Toolkit)

BAIR – Berkeley-evo istraživanje umjetne inteligencije (eng. Berkeley Artificial Intelligence Research)

AWS – Amazon web servisi (eng. Amazon Web Services)

NER – Prepoznavanje imenovanih entiteta (eng. Named-entity recognition)

CoNLL – Konferencija o učenju prirodnog jezika (eng. Conference on Natural Language Learning)

SIGNLL – Specijalna grupa za proučavanje učenja prirodnog jezika (eng. Special Interest Group on Natural Language Learning)

IOB - Inside–outside–beginning

CMD - Command Prompt

**Slike**

[Slika 1 Dijagram procesa dubinske analize podataka 8](#_Toc532138234)

[Slika 2 Primjer HTML koda sa "nebitnim" podacima 10](#_Toc532138235)

[Slika 3 Primjer POS označavanja 11](#_Toc532138236)

[Slika 4 Tehnologije dubinske analize podataka 13](#_Toc532138237)

[Slika 5 Nadzirano učenje 18](#_Toc532138238)

[Slika 6 Nenadzirano učenje 18](#_Toc532138239)

[Slika 7 Primjer nepotpunih/netočnih podataka 19](#_Toc532138240)

[Slika 8 Podjela početnog seta podataka na trening i test set 20](#_Toc532138241)

[Slika 9 Primjer regresijske linije Udjel poreza u BDP u usporedbi s BDP po glavi 21](#_Toc532138242)

[Slika 10 Graf stope netočnosti po vremenu 23](#_Toc532138243)

[Slika 11 Utjecaj stope učenja na stopu netočnosti 25](#_Toc532138244)

[Slika 12 Primjer TensorBoard vizualizacije 27](#_Toc532138245)

[Slika 13 Keras logo 28](#_Toc532138246)

[Slika 14 Amazon Web Services logo 30](#_Toc532138247)

[Slika 15 Mnogobrojan izbor verzija za preuzimanje Python programskoj jezika 34](#_Toc532138248)

[Slika 16 Python instalacijski prozor 34](#_Toc532138249)

[Slika 17 Python u CMD-u 35](#_Toc532138250)

**Programski isječci**

[Kod 1 Učitavanje seta te indeksiranje riječi i NER oznaka u prepareEmbeddings.py 38](#_Toc532226887)

[Kod 2 Pretvaranje riječi u indekse u prepareEmbeddings.py 39](#_Toc532226888)

[Kod 3 Dohvaćanje rečenica u prepareEmbeddings.py 39](#_Toc532226889)

[Kod 4 Generiranje ulaznih i izlaznih vrijednosti u prepareEmbeddings.py 40](#_Toc532226890)

[Kod 5 Spremanje ulaznih i izlaznih podataka u prepareEmbeddings.py 41](#_Toc532226891)

[Kod 6 Učitavanje vrijednosti u liste iz datoteka u prepareEmbeddings.py 41](#_Toc532226892)

*[Kod 7 Parametri modela u model.py](#_Toc532226893)* [41](#_Toc532226893)

*[Kod 8 Funkcija za postavljanje parametara modela u model.py](#_Toc532226894)* [42](#_Toc532226894)

*[Kod 9 Normalizacija znakova u model.py](#_Toc532226895)* [42](#_Toc532226895)

*[Kod 10 Izlazni sloj u model.py](#_Toc532226896)* [43](#_Toc532226896)

*[Kod 11 Kreiranje modela i postavljane parametara u model.py](#_Toc532226897)* [43](#_Toc532226897)

*[Kod 12 Treniranje modela u model.py](#_Toc532226898)* [44](#_Toc532226898)

*[Kod 13 Spremanje modela i podataka o treningu u model.py](#_Toc532226899)* [44](#_Toc532226899)

# Uvod

Dubinska analiza podataka je u današnje vrijeme postala dio svakodnevnice. Posvuda u našoj okolini se nalaze podaci dobiveni dubinskom analizom te ih koristimo kako bi iz velike količine podataka mogli izvući nama potrebne informacije. U ovom diplomskom radu opisati će se proces i tehnologije dubinske analize podataka te proces i tehnologije strojnog učenja.

Proći će se kroz proces dubinske analize podataka. Objasnit će se koraci dubinske analize podataka, od prikupljanja podataka, filtriranja nebitnih podataka, normalizacije do pronalaženja uzoraka i izvlačenja korisnih informacija.

Proučit će se pojam i vrste strojnog učenja te će se ukratko objasniti koraci koji su neophodni u strojnom učenju. Od prikupljanja podataka, prilagodbe tih podataka za strojno učenje, kreiranja modela i treniranja istog, do evaluacije i uglađivanja parametara kako bi krajnji rezultat bio što točniji i precizniji.

Proučit će se i usporediti najpopularnije tehnologije za lokalno strojno učenje te strojno učenje u oblaku. Objasniti prednosti i nedostaci pojedinih tehnologija te objasniti odabir Keras biblioteke za izradu praktičnog djela.

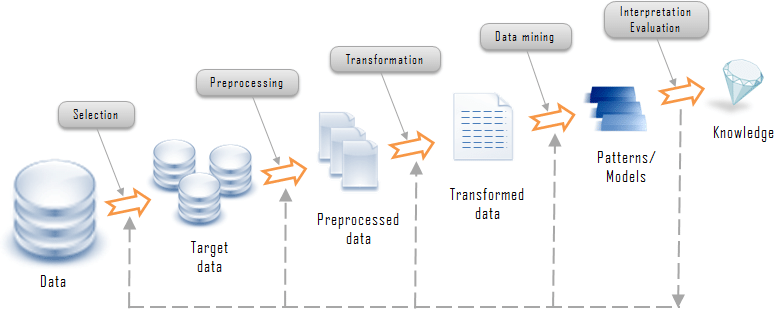
Kao programski jezik za razvijanje praktičnog djela izabran je Python, te se ukratko opisuje Python i njegov proces instalacije, te proces instalacije biblioteka.

Na kraju će se opisati postupak izrade praktičnog dijela ovog diplomskog rada sa objašnjenim primjerima koda i slikama aplikacije.

# Dubinska analiza podataka

Dubinska analiza podataka je proces otkrivanja zanimljivih i korisnih uzoraka i odnosa u velikim količinama podataka. Kombinira znanja iz statistike i umjetne inteligencije sa bazama podataka kako bi analizirali velike digitalne kolekcije podataka. Dubinska analiza podataka usko je povezano sa strojnim učenjem. Podaci dobiveni dubinskom analizom često se koriste kao ulazni set podataka za strojno učenje. Podaci za dubinsku analizu mogu biti bilo što, od komentara i objava sa društvenih mreža, raznih tekstova i književnih djela do običnog niza brojeva.

## Proces dubinske analize podataka



Slika 1 Dijagram procesa dubinske analize podataka

Proces dubinske analize podataka je dugotrajan i ne postoji univerzalno rješenje. Svaki set podataka zahtjeva poseban način obrade, no može se podijeliti na nekoliko segmenata, a to su redom:

* Prikupljanje podataka
* Izdvajanje podataka
* Normalizacija podataka
* Obrada podataka
* Analiza podataka
* Izvlačenje znanja

### Prikupljanje podataka

Proces dubinske analize podataka započinje pronalaženjem odgovarajućeg izvora i prikupljanjem podataka s njega. Što je izvor čišći, tj. što manje netočnih i nebitnih riječi ima, to će proces dubinske analize podataka biti točniji i učinkovitiji. Potrebno obratiti pozornost na privatnost podataka, tj. jesmo li autorizirani koristiti te podatke, te ako jesmo, u koje svrhe ih smijemo koristiti. Neki od mogućih izvora podataka su:

* Publikacije (knjige, časopisi, novine)
* Ankete (telefonske, papirnate, online)
* Internet (web stranice, podatkovni setovi)

Publikacije su najsigurniji izvori podataka, jer su podaci koji se nalaze u knjigama i novinama uglavnom gramatički i pravopisno točni. U takve izvore podataka ulaze i stare knjige koje se digitaliziraju kako bi se očuvala djela autora i kulturna baština te se kasnije javno objavljuju u edukativne svrhe. Najbolji primjer takvih starih digitaliziranih djela se nalazi u NLTK[[1]](#footnote-2) biblioteci u Python programskom jeziku, gdje se nalaze djela poput „Moby Dick“-a.

Ankete su dobri izvori podataka ako se traži nečije mišljenje ili stajalište o nekoj temi. Ispunjavaju ih ljudi te se ti podaci kasnije koriste kako bi se npr. unaprijedila neka usluga ili proizvod. Problem kod takvog izvora podataka je nepotpunost podataka, jer korisnici uglavnom ne ispune anketu na željeni način ili daju lažne podatke što utječe na krajnji rezultat.

Na kraju, Internet je najdostupniji i najlakši izvor podataka, no i najnesigurniji. Nitko ne garantira da su podaci preuzeti sa interneta točni i potpuni. Kao primjer se može pogledati stranica „Wikipedia“, koja je uglavnom glavni izvor podataka i informacija za većinu ljudi. Iako postoje ljudi koji se brinu za ispravnost sadržaja, mogućnost da svaki korisnik upisuje podatke i informacije na stranici ostavlja veliku mogućnost da ti podaci neće biti u potpunosti točni. No postoje razne stranice koje nude gotove setove podataka za razne potrebe, poput „UCI“, „GloVe “, „Google“… Setovi podataka sa takvih izvora su puno sigurniji odabir pošto su podaci uglavnom već formatirani i spremni za čitanje. Te podatke je netko pripremio i obradio kako bi se koristili u svrhu edukacije i raznih istraživanja, te se kasnije javno objavio. Takvi podaci uglavnom sadrže vrlo malo grešaka.

### Izdvajanje podataka

Nakon prikupljanja, potrebno je selektirati željene podatke. Ako je prikupljen gotov set podataka, odnosno podaci su strukturirani, takvi setovi podataka su često multifunkcionalni i mogu se koristiti u različite namjene. Stoga je potrebno iz tog seta izvući potrebne podatke.

Ako pak ne postoji struktura podataka, što je slučaj kod prikupljanja podataka koji nije podatkovni set, tada je potrebno kreirati vlastitu strukturu podataka prema kojoj će se selektirati podaci. Ovaj korak se najviše dotiče sa web stranicama. Preuzimanje podataka s web stranica uključuje preuzimanje HTML[[2]](#footnote-3) koda stranice, te je potrebno izbaciti nebitne elemente iz HTML koda, poput skripti i meta podataka. Potrebno je selektirati samo podatke koji imaju neki značaj.



Slika 2 Primjer HTML koda sa "nebitnim" podacima

### Normalizacija podataka

Mnoge riječi imaju isto značenje, isto se čitaju, no drugačije se pišu. Primjerice, riječ „Coca-Cola“ možemo napisati na nekoliko različitih načina: „Coca-Cola“, „Coca-cola“, „Coca Cola“, „Coca cola“, “Cola“ … Sve te riječi imaju isto značenje, odnosno fonološki sve su riječi iste, no grafološki gledano, možemo primijetiti očigledne razlike u vidu velikog i malog slova, te spojnice (crtica koji spaja dvije riječi). Ovo je sam jedan od primjera gdje normalizacija podataka povećava točnost krajnjeg rezultata. Takvi se podaci često moraju ručno tražiti i ispravljati. Također, u gornjem primjeru možemo vidjeti i razliku velikog i malog slova, čime „Coca cola“ neće biti jednako „Coca Cola“. Taj se problem rješava tako da se sve riječi svedu na mala slova.

Dijakritički znakovi[[3]](#footnote-4) također predstavljaju problem. Na primjer „Luka Modrić“ se često može vidjeti da kao „Luka Modric“, što predstavlja problem. Zahtjeva dodatnu potrošnju vremena pri spremaju podataka tako što se mora paziti na kodiranje (eng. *encoding*) znakova. Ovaj se problem rješava tako da se zamjene dijakritički znakovi sa najbližim srodnim znakom.

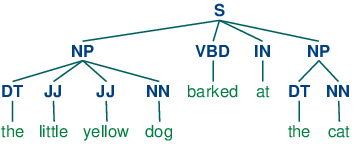
Uklanjanje *stopword*-a[[4]](#footnote-5) još je jedna metoda za normalizaciju podataka. Tako se može ukloniti velik dio podataka koji ne bi imali utjecaj na konačan rezultat, te se time dodatno poboljšati konačni rezultat.

### Obrada podataka

Nakon filtriranja i normalizacije, dolazi obrada podataka. Podaci se obrađuju ovisno o svrsi u koju se koriste. Pošto se u ovom diplomskom radu govori o generiranju pitanja, fokus će biti na obradi potrebnoj za tu svrhu. Kao ulazni podatak će se uzeti velika količina teksta.

Za početak, tekst je potrebno podijeliti na rečenice. Zvuči kao vrlo jednostavan zadatak, tražimo rečenične znakove koji označavaju kraj rečenice, no i nije sve tako jednostavno. Jedan od problema su kratice, koje se često pišu s točkom. Jedno od rješenja može biti kreiranje liste kratica te ignoriranje tih riječi. Također, postoje razne biblioteke, poput već spomenutog NLTK, koje će takve stvari odraditi.

Rečenice je zatim potrebno podijeliti na riječi, koje se zatim označavaju (eng. *tagging*) POS (eng. *Part-of-speech*) oznakama. Riječi se označavaju po tome koja su vrsta riječi (imenica, glagol,…). Tako se dobivaju odnosi između riječi u rečenici.



Slika 3 Primjer POS označavanja

### Analiza podataka

Obrađeni podaci sami po sebi mogu dati određeno podatke, no oni uglavnom nisu traženi podaci, već je potrebna analiza podataka. Analiza se odrađuje primjenom jedne ili više tehnika za dubinsku obradu podataka. Neke od tehnika su:

* Traženje uzoraka (eng. Tracking patterns) – osnovna tehnika dubinske analize podataka. Traže se ponavljajuće vrijednosti ili težnja podataka prema nekim vrijednostima.
* Klasifikacija (eng. Classification) – kompleksna tehnika dubinske analize podataka koja određuje kategoriju ili klasu podatka prema ulaznom setu klasa koje je korisnik odredio.
* Asocijacija (eng. Association) – slična tehnika kao traženje uzoraka, no ovdje se traže podaci koji su u korelaciji sa drugim podacima
* Raspoređivanje u razrede (eng. Clustering) – tehnika slična klasifikaciji, no omogućuje dodatno raspoređivanje podataka na temelju njihovih sličnosti
* Regresija (eng. Regression) – tehnika za određivanje vjerojatne vrijednosti nekog podatka na temelju drugih podataka.
* Predviđanje (eng. Prediction) – jedna od najvažnijih tehnika dubinske analize podataka. Koristi se za predviđanje budućih podataka na temelju ulaznih podataka. Povezuje ponašanje podataka sa određenim događajima u okolini.

### Izvlačenje znanja

Analizom podataka dobiveni su novi podaci, uzorci i setovi podataka koji se mogu vizualizirati za prezentacijske svrhe, transformirati za aplikacijske svrhe, te se mogu ukloniti redundantni podaci, ako nam nisu bitni.

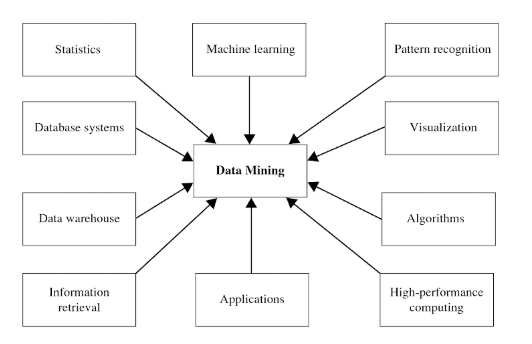
Kreiranje izvještaja još je jedna korisna akcija. Kreiranje izvještaja o dubinskoj analizi podataka omogućuje da se zabilježe svi koraci i rezultati. Ti se izvještaju mogu koristiti u budućim analizama kako bi se, ako je moguće, poboljšala točnost, performanse i slično.

## Tehnologije dubinske analize podataka

Dubinska analiza podataka je integracija više tehnologija, a neke od bitnijih tehnologija su:

* Statistika
* Strojno učenje
* Baze i skladišta podataka
* Izvlačenje informacija

Uglavnom se ne koristi samo jedna, nego kombinacija tih tehnologija.



Slika 4 Tehnologije dubinske analize podataka

### Statistika

Statistika proučava prikupljanje, analizu, interpretaciju i prezentaciju podataka te doprinosi dubinskoj analizi podataka statističkim modelom, setom matematičkih funkcija koje opisuju ponašanje objekata u određenoj klasi na temelju nasumičnih varijabli i njihovih pridruženih raspodjela vrijednosti. Koriste se kako bi se kreirao model podataka i klasa te mogu biti rezultat dubinske analize podataka, isto kao što može biti i ulazni set za dubinsku analizu podataka. Primjerice, može se kreirati statistički model za oštećene (eng. *corrupted*) i nedostajuće podatke te pomoći otkriti i sanirati te podatke.

### Strojno učenje

Dubinska analiza podataka je jako blizak pojam sa strojnim učenjem. Glavna zamisao je da se trenira model raznim pravilima uzorcima dobivenim dubinskom analizom te primjena tih pravila na rješavanje problema. Kombinira se sa statističkim tehnikama kako bi se razvile sofisticirane statističke tehnike za dubinsku analizu podataka.

### Baze i skladišta podataka

Baze podataka su organizirane kolekcije podataka. Kako je za dubinsku analizu podataka potrebna velika količina ulaznih podataka, baze podataka su idealno rješenje, zbog svoje optimizacije, skalabilnosti i sposobnosti za procesuiranje velikih količina podataka. Također, dubinska analiza podataka se može koristiti u bazama podataka kako bi proširile mogućnosti koje baze podataka nude. Skladišta podataka mogu sadržavati podatke iz više iz više različitih izvora, te se koriste za analizu podataka i kreiranje izvještaja.

### Izvlačenje informacija

Izvlačenje informacija (eng. *Information retrievel,* IR) je znanost traženja dokumenata nestrukturirane prirode (uglavnom tekst ili multimedija) koji zadovoljava informacija u velikoj kolekciji (bazi podataka na računalu). Razlika između baza podataka i izvlačenja informacija je u dvije stvari:

* Baze očekuju strukturirane ulazne podatke
* Upiti su formirani po ključnim riječima iz teksta, ne u nekom jeziku (SQL[[5]](#footnote-6))

Tipično izvlačenje informacija uključuje model vjerojatnosti, u kojem se ulazni tekstni dokument prezentira kao set riječi koje kreiraju jezični model i gustoći određenih riječi u tekstu. Sličnost između dva dokumenta je rezultat sličnosti njihovih modela.

## Primjene dubinske analize podataka

Dubinska analize podataka može se koristiti u mnogim područjima, od proizvodnje i marketinga, medicinskih istraživanja, istraživanja prijevara do analize tržišta za velike trgovačke lance.

### Dubinska analiza teksta

Dubinske analize podataka se može primijeniti na različite vrste tekstualnih dokumenata, koji predstavljaju nestrukturirane podatke, kako bi se klasificirali članci, knjige, dokumenti, elektronička pošta i web stranice. Dobar primjer su web stranice, koje sadrže, osim teksta, i različite HTML elemente (<div>, <body>, …) i entitete (&nbsp;, &euro;…). Razne online aplikacije (email klijenti) već imaju ugrađene filtere neželjene („*spam*“) pošte koji rade na principu dubinske analize podataka teksta.

### Medicina

Iskoristivost dubinske analize podataka je neprocjenjiva u medicini. Omogućuje rano otkrivanje bolesti na temelju kliničkih testova i istraživanja, smanjuje troškove liječenja te poboljšava skrb za pacijenta.

### Procjena rizika

Analiza rizika je vrlo bitan segment svih velikih događanja, putovanja, ulaganja. Na primjer, turističke agencije koriste procjenu rizika kako bi svojim putnicima priuštili sigurna putovanja. To rade na način da analiziraju podatke o broju kriminalnih djela, vremenskim uvjetima, prirodnim katastrofama i slično te tako rangiraju turističke destinacije.

### Marketing

Dubinska analiza podataka na području marketinga uvelike je poboljšala rezultate marketinških kampanja. Neki od bitnijih segmenata u području marketinga su analiza tržišta, analiza potrošačke košarice, predviđanje stope pozitivne reakcije na marketinške kampanje, analiza ponašanja kupaca u trgovini i sl. Također, danas je vrlo popularna analiza povijesti pretraživanja interneta preko koje se korisniku prikazuju reklame u ovisnosti o pojmovima koje je pretraživao.

# Proces strojnog učenja

U ovom poglavlju će se proći kroz proces strojnog učenja. No da bi razumjeli proces strojnog učenja, prvo valja razumjeti pojam strojnog učenja.

## Strojno učenje

Da bi razumjeli što je strojno učenje, prvo je potrebno znati sljedeće pojmove: stroj, inteligencija, umjetna inteligencija te učenje. Stroj (ili uređaj) jest skup dijelova povezanih u jednu logičnu cjelinu s ciljem izvođenja određene operacije. Za primjer strojnog učenja, kao stroj se može uzeti računalo. Računalo samo po sebi nema sposobnost samostalnog donošenja odluka, iako su računala u današnje vrijeme došla vrlo blizu tom koraku, odnosno podarena im je umjetna inteligencija.

Pojam inteligencije se klasificira kao mentalna karakteristika koja se sastoji od sposobnosti za učenje iz iskustva, prilagodbe na nove situacije, razumijevanja i korištenja apstraktnih pojmova i korištenja prethodnih znanja za snalaženje u novoj okolini u kojima ne pomaže stereotipno nagonsko ponašanje. Prema definiciji, primjena stečenih znanja i iskustava u novim okolinama je glavna karakteristika inteligencije. Ako inteligenciju promatramo sa stajališta računala, možemo je objasniti na primjeru prepoznavanja slika, odnosno objekata na slikama. Testnom subjektu predočena je određena količina uzoraka nekog objekta, npr. slike mačaka, te nakon određenog perioda taj bi subjekt trebao imati mogućnost prepoznati traženi objekt na novom uzorku, nekom iz kojeg nije učio te ga prvi puta vidi. Kada računalo ima tu mogućnost, onda kažemo da posjeduje umjetnu inteligenciju. Umjetna inteligencija (UI) je naziv koji pridajemo svakom neživom sustavu koji pokazuje sposobnost snalaženja u novim situacijama (inteligenciju). Inteligencija se stječe učenjem novih pojmova i znanja te je učenje ključno za stjecanje i povećanje inteligencije.

Definicija učenja, prema rječniku, je:

* Dobivanje znanja učenjem, iskustvom
* Učenje preko informacija ili promatranjem
* Pamćenje
* Dobivanje instrukcija
* Utvrđivanje

Ta definicija vrijedi za ljude, dok za računala vrijede samo zadnje tri točke. Prve dvije točke ne mogu biti primijenjene na računala jer je nemoguće utvrditi da li je računalo samostalno prikupilo informacije i naučilo, ne možemo ga pitati ili očitati informacije. Zadnje tri točke su dosta trivijalne i jednostavne i na takav način se mogu izvršiti na računalu.

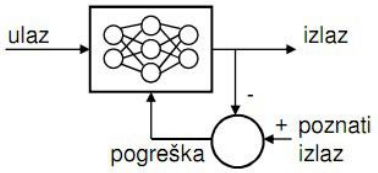
Strojno učenje je grana umjetne inteligencije koje često koristi statističke tehnike kako bi računalu dalo mogućnost da uči iz podataka, bez da je eksplicitno programirano. Strojno učenje se temelji na algoritmima koji imaju mogućnost učiti i dati predikciju rezultata temeljenu na nekoj vrsti podataka. Takvi algoritmi daju rezultate temeljene na prijašnjim ulaznim podacima te balansiraju svoje parametre ovisno o novim ulazima, ne slijede strogo zadana pravila. Dobar primjer bio bi email filter, koji ima početni ulazni set podataka i ključne riječi koje traži u tekstu, te kasnije prilagođava svoj korpus ovisno o korisničkim unosima. Nove email-ove, koji su označeni kao „*spam*“, provlači kroz razne algoritme i uspoređuje s drugima te tako pronalazi nove ključne riječi i dodaje ih u svoj korpus. Iako takvi algoritmi rade sa velikom točnošću, i dalje nisu savršeni i mogu pogriješiti.

## Vrste strojnog učenja

Postoji nekoliko vrsta strojnog učenja:

* Nadzirano učenje
  + Polu-nadzirano učenje
  + Aktivno učenje
  + Učenje pojačavanjem
* Nenadzirano učenje

U nadziranom učenju dostupan je skup uzoraka za učenje i za svaki uzorak je poznat rezultat. Tijekom procesa učenja izvode se iteracije ponavljanja istih uzoraka sve dok izlaz ne odgovara očekivanim rezultatima.

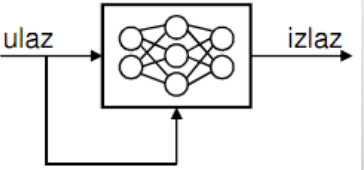


Slika 5 Nadzirano učenje

Polu-nadzirano učenje se temelji na nadziranom učenju, no uzorci često nisu potpuni, odnosno svi uzorci za učenje sadrže ulazne parametre, ali samo neki sadrže i izlazne.

Aktivno učenje je specijalan slučaj polu-nadziranog učenja u kojem algoritam zahtjeva unose korisnika za određeni ulaz kako bi se dobio željeni izlaz za nove uzorke.

Nenadzirano učenje je slično kao i nadzirano učenje, no za skup uzoraka ne postoje poznati rezultati koji bi se uspoređivali sa izlazima. Tijekom učenja mreža na različite ulazne uzorke reagira na različite načine te tako stvara internu reprezentaciju ulaznih podataka. Ova vrsta učenja se najčešće koristi kod grupiranja podataka te prepoznavanja sličnosti.



Slika 6 Nenadzirano učenje

Učenje pojačavanjem je hibridni način koji uključuje aspekte nadziranog i nenadziranog učenja. Neuronska mreža prima uzorke za učenje za koje nema očekivanog rezultata, ali je za svaki izlaz poznato je li on ispravan ili neispravan za zadani ulaz, što je postignuto povratnom vezom.

## Koraci strojnog učenja

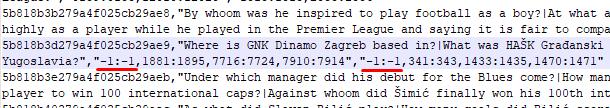
Strojno učenje je grana umjetne inteligencije koja se još uvijek uvelike razvija i otkrivaju se novi i efikasniji načini kako bi proces bio što učinkovitiji. No, iako se strojno učenje još uvijek razvija, postoje određeni koraci koji će se uvijek morati slijediti, a to su redom:

* Prikupljanje podataka
* Priprema i prilagodba podataka za strojno učenje
* Kreiranje modela
* Treniranje modela
* Evaluacija modela
* Mijenjanje parametara ovisno o rezultatima

U daljnjem tekstu se opisuje proces nadziranog strojnog učenja. Pošto je prikupljanje podataka slično kao i kod dubinske analize podataka, taj se dio preskače.

### Priprema i prilagodba podataka za strojno učenje

Podaci prikupljeni sa bilo kojeg izvora nisu dobar ulazni set za strojno učenje, potrebno ih je prilagoditi kako bi ih algoritam mogao razumjeti. Za početak, može se proći kroz sve podatke i ukloniti duplikate, jer dupli podaci u setu podataka stvaraju samo smetnju i mogu imati negativan utjecaj na rezultate strojnog učenja. Zatim je potrebno ispraviti greške i nedostatke u podacima. Ako podaci ne sadrže neke vrijednost, tj. nepotpuni su, može doći do grešaka u algoritmu za učenje ili algoritam može naučiti krive vrijednosti, stoga je te podatke potrebno ispraviti ili izbaciti kako bi ubrzao i unaprijedio proces učenja. No isto tako je moguće da te vrijednosti neće imati značajan utjecaj na proces učenja, stoga se mogu zanemariti. Vrijednosti koje nedostaju su uglavnom označene sa numeričkom vrijednosti (-1, 0), praznim mjestom ili '-' znakom.

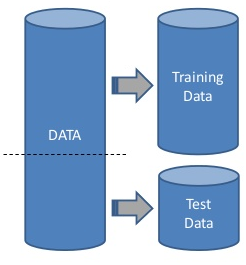


Slika 7 Primjer nepotpunih/netočnih podataka

Nakon uklanjanja duplikata i grešaka, potrebno je pretvoriti podatke u računalu razumljive podatke i normalizirati vrijednosti. To ukratko znači da bi svaki podatak iz ulaznog seta podataka trebao imat jedinstvenu brojčanu vrijednost kako bi ih algoritam za strojno učenje mogao razlikovati. Postoje razna gotova rješenja poput GloVe, globalni vektor za reprezentaciju riječi (eng. Global Vectors for Word Representation). Jedno jednostavno rješenje je dodavanje svih jedinstvenih riječi u listu pa je brojčana reprezentacija riječi indeks elementa u listi.

#### Kreiranje seta podataka

Daljnja prilagodba podataka odvija se tako da se podaci dijele u setove. Jedan se set podataka koristi za učenje i formiranje modela, odnosno treniranje, trening set, dok se drugi set podataka koristi za test modela, testni set. Trening set podataka trebao bi sadržavati sve vrijednosti, odnosno ulazne i izlazne podatke, dok bi testni set podatka trebao sadržavat samo ulazne podatke. Testni set podataka ne bi se trebao mijenjati sve dok se ne kreira najbolji model. Koristi se za evaluaciju modela kao set nepoznatih i još ne viđenih podataka. Omjer tih setova u odnosu na sveukupni set podatka je uglavnom 7:3 ili 8:2. Dobra je praksa nasumično spremati podatke u setove podataka, kako bi se izbjeglo „navikavanje“ algoritma na određene vrijednosti.



Slika 8 Podjela početnog seta podataka na trening i test set

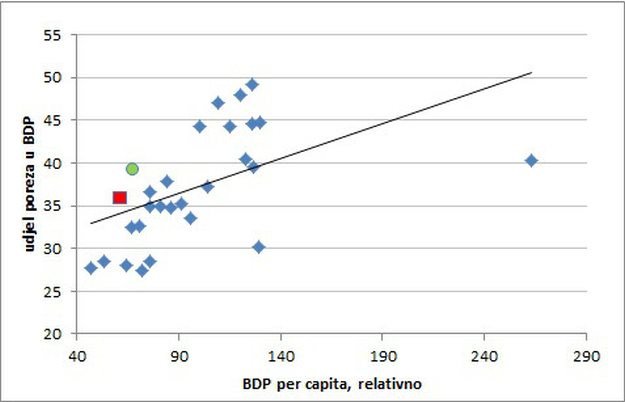
### Kreiranje modela

Kreiranje modela ovisi o ulaznim podacima. Model se kreira prema parametrima ulaznih podataka, odnosno mora imati onoliko ulaza koliko je ulaznih podataka, te isto tako i sa izlazima. Između ulaznog i izlaznog sloja se nalaze slojevi koji sadrže neurone čije vrijednosti zapravo određuju model. Također, moguće je kombinirati više modela tako da izlaz jednog bude ulaz drugog. Postoji mnogo gotovih modela koje su znanstvenici kreirali pomoću svojih istraživanja koji se vrlo lako mogu pronaći i upotrijebiti. Primjerice, postoje gotovi modeli za slikovne tipove podataka, tekstualne, numeričke i sl. Neki od poznatijih algoritama za učenje su:

* Linearna regresija (engleski. *Linear Regression*)
* Stabla odluke (engleski. *Decision Trees*)
* Naivna Bayes-ova klasifikacija (engleski. *Naive Bayes Classification*)

#### Linearna regresija

Linearna regresija je idealna kada su svi podaci numerički. Ovaj algoritam ostvaruje vezu između ulaznih i izlaznih vrijednosti preko najboljeg pravca, koji se još naziva regresijska linija. Najčešće se koristi za procjenu rasta i pada cijena nekretnina, dionica i sl. Vrlo je dobar izbor za statističke procjene, no problem je ako podaci pokazuju nelinearnu zavisnost, tada se regresijska linija nikada neće pronaći.



Slika 9 Primjer regresijske linije Udjel poreza u BDP u usporedbi s BDP po glavi

#### Stabla odluke

Stabla odluke, kao i svugdje, rade na principu podjeli pa vladaj. U ovom algoritmu se vrijednosti dijele na dvije ili više homogenih setova. Mora se pronaći najčišći ulazni atribut koji će se pozicionirati kao korijen stabla. Mjera za čistoću ulaznog atributa je informacija i mjeri se u bitovima. Informacija predstavlja očekivanu količinu informacija koja je potrebna da bi se odredilo da li je novi ulazni podatak klasificiran kao istina ili laž. Računa se prema sljedećoj formuli:

A predstavlja atribut na koji se ulazni set dijeli, S je ulazni set podataka, T je set podataka kreiran razdvajanjem atributa A od seta S, p(t) je udio broja elemenata t u setu S. H(S) je entropija seta S, koja predstavlja čistoću podataka u proizvoljnoj kolekciji podataka, dok je H(t) entropija seta t. Entropija se računa prema sljedećoj formuli:

gdje je S je ulazni set podataka, p(c) udio elemenata klase c u setu S. Na kraju se odabire atribut kojem je vrijednost informacije najveća. Ista stvar se ponavlja sa ostale atribute dok se stablo potpuno ne složi.

Stabla odluke se najčešće koriste za klasifikaciju problema. Za razliku od linearnih algoritama, nemaju problema sa nelinearnim vrijednostima.

#### Naivna Bayes-ova klasifikacija

Temeljena na Bayes-ovom teoremu, ova tehnika klasifikacije se bazira na nepovezanosti atributa ulaznog seta podataka. Primjerice, jabuka je okrugla, crvena i veličine oko 10 centimetara u promjeru. Iako te vrijednosti zavise jedna o drugoj, svaka od njih zasebno može doprinijeti odluci algoritma da je ulazni podatak zapravo jabuka, te se zbog toga ova klasifikacija naziva naivnom.

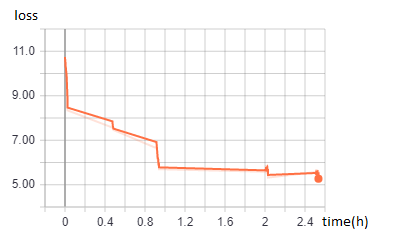
Naivna Bayes-ova klasifikacija idealna je za velike setove podataka. Poznata je po svojoj jednostavnosti te dobrim performansama čak i sa vrlo kompleksnim problemima. Izlazna vjerojatnost se računa prema formuli:

gdje je x ulazni podatak, c je atribut podatka, P(c | x) je izlazna vjerojatnost za ulazni podatak c, P(c) udio ulaznog atributa u ukupnom setu podataka dok je P(x) je udio ulaznog podatka u ukupnom setu podataka.

Zbog svoje brzine, ovaj se algoritam često koristi za predviđanje u realnom vremenu. Najčešće se koristi za klasifikaciju teksta i kao „*spam*“ filter za detektiranje neželjenih poruka i email-ova.

### Treniranje modela

Za treniranje modela koristi se trening set podataka. Treniranje se obavlja tako da se ulazni podaci iz trening seta daju modelu da prilagodi svoje parametre kako bi izlazi odgovarali onima iz trening seta. Krajnji cilj treninga je da model točno predvidi izlaze sa što većom točnošću. Ovo je ujedno i najduži dio procesa strojnog učenja. Ukoliko se koriste veliki setovi podataka, treniranje modela može trajati satima, danima ili čak mjesecima. Treniranje modela završava ili određenim brojem ponavljanja ili kada stopa netočnosti padne ispod određenog broja. Ta ponavljanja se nazivaju epohe. Jedna epoha je jedan prolaz ulaznog seta podataka kroz model, tj. epoha završava kada svi podaci iz ulaznog seta prođu kroz model. Stopa netočnosti je interpretacija koliko dobro je model prilagodio svoje parametre.



Slika 10 Graf stope netočnosti po vremenu

### Evaluacija modela

Nakon treninga modala dolazi evaluacija da se vidi da model daje rezultate prihvatljive točnosti. Ovaj bi korak trebao biti reprezentacija performansi modela u stvarnom svijetu. Evaluacija modela se obavlja na više načina. Uglavnom se koriste neke metrike kao pokazatelji performansi modela.

Kao prvi način se može gledati stopa netočnosti koja je najbolji pokazatelj koliko su dobro postavljeni parametri modela. Što je niža stopa netočnosti, to je bolji model, osim u slučajevima kada je trening set previše tematski homogen, što može dovesti do „navikavanja“ modela na određene podatke, te loše performanse sa drugim podacima.

Drugi način je evaluacija uspješnosti modela na dosad neviđenim podacima. U tu svrhu se kreira evaluacijski set podataka koji nije tematski povezan sa trening ili test setom. Na taj se način zapravo najbolje testira mreža jer su ti podaci stvarno „neviđeni“, dok je testni set podataka također neviđen, ali je tematski vezan za trening set, iako se evaluacija može obaviti i sa testnim setom, ako je tematski heterogen.

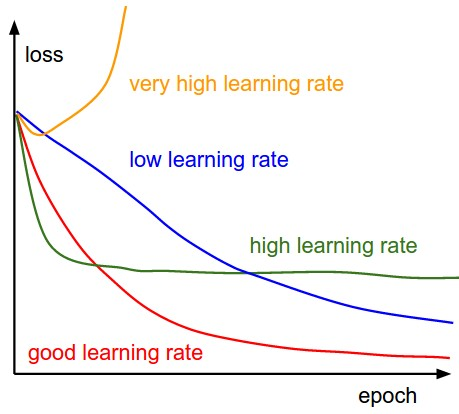
U konačnici, ako evaluacija modela zadovolji kriterije, model je spreman za stvarni svijet. No u većini slučajeva, odnosno gotovo nikad, prvi model nije idealan te zahtjeva male ili velike preinake kako bi rezultati bili što bolji.

### Mijenjanje parametara ovisno o rezultatima

Mijenjanje parametara ovisi o rezultatima evaluacije modela. Što su rezultati evaluacije bolji, to se manje promjena mora raditi. Parametri se mijenjaju kako bi se pokušalo dobiti što bolje rezultate u bilo kojem smislu, ili brži trening, ili veću točnost. Parametri koji se mogu mijenjati kako bi se poboljšali rezultati su sljedeći:

* Stopa učenja
* Model
* Algoritam za učenje
* Ulazni set podataka

Ako je model blizu stanja koje tražimo, onda je potrebno fino ugađanje parametara koje se postiže mijenjanjem stope učenja. Stopa učenja je parametar koji kontrolira koliko brzo će algoritam mijenjati svoje vrijednosti u svakom koraku. Što je taj iznos veći, to će algoritam brže učiti, no ako je iznos stope učenja prevelik, može doći do izobličenja vrijednosti te postupak učenja može otići u krivom smjeru. U slučaju da je stopa učenja premalena, algoritam radi premalene promjene i presporo uči te je negativna posljedica to što proces učenja može predugo trajati. Također, mijenjanje stope učenja može pomoći ukoliko je stopa netočnosti zapela na određenoj vrijednosti. Stoga treba testirati sa raznim vrijednostima i vidjeti koja najbolje odgovara. Uz stopu učenja, mogu se mijenjati i drugi parametri algoritma koji mogu imati utjecaj na proces učenja, koji se mogu pronaći u dokumentaciji algoritma.



Slika 11 Utjecaj stope učenja na stopu netočnosti

Ako mijenjanjem stope učenja i drugih parametara nije uočena veća promjena u rezultatima, vrijeme je za uređivanje modela. Na modelu se rade sitne preinake kako bi se poboljšali rezultati poput dodavanja/oduzimanja slojeva neurona ili mijenjanja postojećih.

Mijenjanje algoritma za učenje malo je drastičniji korak. Pri promjeni algoritma treba voditi računa da odgovara našem modelu, odnosno da može raditi sa tipom podataka kojim mi raspolažemo. Nisu svi algoritmi za sve tipove podataka. Alternativni algoritmi kao i njihova upotreba mogu se pronaći u dokumentaciji programskog jezik algoritma.

Ukoliko niti jedan od prijašnjih koraka nije donio poboljšanje rezultata, moguće je da ulazni set podataka nije pravilno procesuiran i očišćen od nepotpunih i netočnih podataka ili su podaci tematski previše slični i mreža se previše „navikla“ na takve podatke.

# Tehnologije strojnog učenja

Već je zaključeno da je strojno učenje vrlo popularna grana umjetne inteligencije, stoga ne čudi interes mnogih velikih kompanija, poput Microsoft-a, Google-a, Amazon-a i sl. Svatko bi želio svoj udio u tehnologiji budućnosti. Velike kompanije nude vrlo popularna rješenja strojnog učenja u oblaku (engleski cloud). Osim rješenja u oblaku, postoje i biblioteke i programski okviri (engleski Framework) koje se izvršavaju na lokalnom računalu.

## Lokalne tehnologije strojnog učenja

Lokalne tehnologije zahtijevaju računalo na kojem će se izvršavati te biblioteke i programski okviri. Također, potrebno je samostalno napisati programski kod u nekom od programskih jezika za iskorištavanje tih biblioteka. Neke od poznatijih biblioteka i programskih okvira su:

* TensorFlow
* Keras
* Caffe

### TensorFlow

TensorFlow je biblioteka otvorenog koda visokih performansi za numeričke obračune. Razvili su je Google-ovi stručnjaci iz „Google Brain“ tima 2015. godine unutar Google-ove AI organizacije. TensorFlow je jedan od najpopularnijih i najraširenijih sustava za umjetnu inteligenciju i strojno učenje.

Arhitektura TensorFlow-a omogućava korisnicima razvoj i računanje na različitim platformama, od korištenja procesorske snage i snage grafičkih kartica do posebnih čipova za akceleraciju procesa strojnog učenja. Nove verzije TensorFlow-a omogućavaju integraciju umjetne inteligencije na mobilne uređaje, uz klasična stolna računala i servere, te web stranice zbog podrške za JavaScript, programski jezik koji se najčešće koristi na Web stranicama. Uz Python, kao standardni programski jezik za TensorFlow, podržani su još i Java, Go i C.

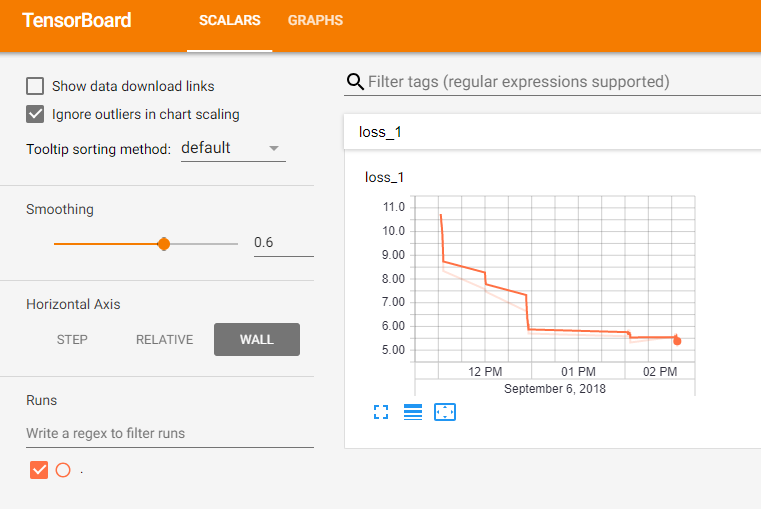
Kako bi se lakše razumjeli, otklonili nepravilnosti (eng. *debug*) i optimizirali programi I modeli kreirani u TensorFlow-u, kreiran je alat za vizualizaciju nazvan TensorBoard. TensorBoard se pokreće preko komandne linije sljedećom komandom:

tensorboard --logdir=putanja\_do\_direktorija

te se predaje parametar „–logdir“ koji je putanja do log direktorija TensorFlow algoritma. Nakon pokretanja, TensorBoard vizualizacija se otvara u Internet pregledniku preko adrese:

localhost:6006 ili ime\_računala:6006

TensorBoard se može koristiti za vizualizaciju grafa stope netočnosti, grafički prikaz modela i sl.



Slika 12 Primjer TensorBoard vizualizacije

### Keras

Kao i TensorFlow, Keras je također biblioteka otvorenog koda izdana 2015. godine. Kreiran je razvijen kao dopuna drugim bibliotekama i programskim okvirima TensorFlow, Theano i CNTK. Ukoliko se eksplicitno ne navede programski okvir, koristit će se TensorFlow. No moguće je koristiti i druge biblioteke, te također napraviti vlastitu. Promjena biblioteke se radi preko konfiguracijske datoteke koja se nalazi u korisničkom direktoriju „ %USERPROFILE%/.keras/keras.json“ na Windows operativnom sistemu ili „$HOME/.keras/keras.json“ na Linux operativnim sistemima. Ta konfiguracijska datoteka izgleda ovako:

{

"image\_data\_format": "channels\_last",

"epsilon": 1e-07,

"floatx": "float32",

"backend": "tensorflow"

}

U „backend“ varijablu upisujemo ime željene biblioteke.

Keras se fokusira na korisničkoj interpretaciji, modularnosti i proširivosti. Kreiran je sa fokusom da omogući što kraći proces od ideje do rezultata.

Minimaliziran je broj korisničkih akcija za učestale slučajeve te vraća razumljive i poduzetne povratne informacije u slučaju grešaka. Slično kao i TensorFlow, može koristiti procesorsku snagu i snagu grafičkih kartica.



Slika 13 Keras logo

### Caffe

Službeno izdan 2017. godine, Caffe (eng. *Convolutional Architecture for Fast Feature Embedding*) je programski okvir za strojno učenje koji se fokusira na brzinu i modularnost. Kao i prijašnji algoritmi, također je otvorenog koda, napisan u C++ sa Python sučeljem. Caffe svoju popularnost najviše može zahvaliti zajednici koja zajedno sa Berkeley AI Research (BAIR), istraživačkim laboratorijem za umjetnu inteligenciju, koji je razvio Caffe, i dalje razvija ovaj programski okvir.

Caffe se najviše koristi za klasifikaciju i segmentaciju slika, zvuka i ostalih multimedija. Na stranici <http://demo.caffe.berkeleyvision.org/> postoji i demo projekt koji reprezentira mogućnost Caffe programskog okvira.

### Usporedba biblioteka i programskih okvira

TensorFlow je daleko najpopularnija biblioteka od svih te ima najveću zajednicu. Zbog konstantnog razvijanja, TensorFlow ima budućnost ispred sebe i novi korisnici dolaze svakodnevno. Postoji i mnogo kvalitetne dokumentacije i literature. Jako je širok obujam problema koje TensorFlow može riješiti, te ima jako dobar vizualizacijski alat TensorBoard. Također, dostupan je i na mobilnim uređajima te na Webu.

Jednostavnost je glavna značajka Keras-a. Postoji velika podrška zajednice i dostupno je mnogo gotovih rješenja. Zbog svoje jednostavnosti i sučelja idealan je za učenje i početnike u strojnom učenju. Loša stvar je to što ima slabu podršku za treninge velikih obujama koji se obavljaju na više računala, a i zbog jednostavnosti gubi na fleksibilnosti, odnosno podržava manji broj postavki od TensorFlow-a. Postoji i deficit u performansama u odnosu na TensorFlow.

Caffe je idealan za velike količine slikovnih podataka za slikovnu klasifikaciju, gdje je još uvijek bolji od TensorFlow-a. Također, postoji velik broj gotovih modela za računalnu viziju koje je moguće preuzeti i iskoristiti. Problemi nastaju kada treba dodati novi sloj u model, pošto je biblioteka napisana u C++, potrebno je mijenjati stvari. Isto tako, nije idealno za stvarni svijet pošto je još u razvoju.

Konačni odabir za izradu praktičnog djela pada na Keras, koji je idealan izbor za ovu vrstu problema- Za izradu ovog rada nisu potrebne napredne mogućnosti koje TensorFlow nudi, a Caffe ne daje pouzdane rezultate u obradi teksta.

## Tehnologije strojnog učenja u oblaku

Rješenja u oblaku u današnje vrijeme uzimaju velik dio kolača. Mnogi poslužitelji ove usluge nude gotove modele ili vrlo intuitivno kreiranje vlastitog modela uz podršku te veliku procesnu snagu, što ubrzava proces treniranja modela. Postoje i mnoga gotova rješenja koja se mogu iskoristiti. Neki od najpoznatijih tehnologija strojnog učenja u oblaku su:

* Amazon Web Services
* Google Cloud
* Azure Machine Learning Studio

### Amazon Web Services

Amazon je američka Internet trgovina osnovana 1994. godine te spada u najveće i najpoznatije svjetske Internet trgovine. No nije Internet trgovina jedina djelatnost kojom se Amazon bavi, posjeduju filmski i televizijski studio, Amazon Studios, svoju liniju tableta i televizora, te najveću svjetsku infrastrukturu u oblaku.

Amazon Web Services (AWS) nudi najširi izbor usluga strojnog učenja i umjetne inteligencije. Korisnici mogu birati između gotovih modela za računalni vid (eng. *computer vision*), jezik, preporuke i predviđanja. Kreiran je alat koji pomaže korisnicima u razvijanju nazvan Amazon SageMaker, koji sadrži algoritme koji ubrzavaju razvoj modela do 10 puta u odnosu na ostala rješenja, nudi uštede korisnicima velikom efikasnošću te je cijeli proces strojnog učenja maksimalno pojednostavljen.

Brojke o Amazon Web Services:

* 85% TensorFlow projekata su razvijeni na AWS platformi
* 81% projekata dubinske analize podataka rade na AWS platformi
* Preko 50% kraće vrijeme treniranja



Slika 14 Amazon Web Services logo

### Google Cloud

Google je jedna od najpoznatijih svjetskih kompanija osnovana 1998. godine. Osim što su vlasnici najpoznatije svjetske tražilice, bave se proizvodnjom softvera i hardvera, online reklamnim tehnologijama te tehnologijama u oblaku. Neka od poznatih softverskih rješenja su Google Maps, Docs, Sheets, Gmail i slično. Vlasnici su popularnog operativnog sustava za mobilne uređaje, Android, najpoznatijeg Internet preglednika, Google Chrome, te web stranice za dijeljenje videa, YouTube.

Google Cloud nudi velik raspon usluga, od skladištenja podataka do strojnog učenja, usluge prevođenja teksta i pretvaranje teksta u govor i obrnuto. Mnogo je i raznih aplikacija koje omogućuju kreiranje naprednih i sofisticiranih te njihovo treniranje, a neke od njih su AI Hub, Cloud TPU, Cloud Natural Language i slično. Google Cloud AutoML je ekvivalent Amazon SageMaker te omogućuje i generiranje visokokvalitetnih podatkovnih setova.

### Azure Machine Learning Studio

Azure je najnovija tehnologija na ovoj listi, razvijena 2010. godine. Njen vlasnik je Microsoft, američka multinacionalna tehnološka kompanija osnovana 1975. Najpoznatiji su po svome operativnom sistemu „Windows“, no bave se i razvojem softvera i hardvera, licenci i povezanih servisa. Najpoznatiji proizvodi, uz Windows operativni sustav, su Microsoft Office, Internet explorer te Visual Studio.

Azure Machine Learning Studio, kao i prijašnje tehnologije, nudi mnoga rješenja za strojno učenje, te mnoga druga područja poput baza podataka, integracije, pohrane podataka i slično. Omogućuju korisniku da integrira svoje programske isječke te koristi neke od mnogih dostupnih biblioteka.

### Usporedba tehnologija za strojno učenje u oblaku

Ne postoji trenutno najbolji izbor tehnologije za strojno učenje u oblaku za svaki problem. Svaka od prije navedenih tehnologija ima svoje mane i prednosti, funkcionira bolje u određenim područjima. No treba napomenuti da je od spomenutih AWS najpoznatija i najskuplja opcija, dok Azure ML Studio nudi ipak prihvatljivije ponude.

# Prepoznavanje imenovanih entiteta

Prepoznavanje imenovanih entiteta (eng. Named-entity recognition) je tehnika korištena u izvlačenju informacija (eng. Information retrieval, IR) kako bi se pronašli i imenovali entiteti u tekstu te klasificirali u predodređene klase. Neki od entiteta koji se pronalaze u tekstu su osobe, lokacije, organizacije, datumi i vremena.

NER se razvija ponajviše za engleski jezik jer je najkorišteniji svjetski jezik. Problematika kod drugih jezika su različite vrste riječi za tekst istog značenja, što je zahtijevalo dodatno procesuiranje. Također, dijakritički znakovi predstavljaju problem pri procesuiranju i zahtijevaju dodatne podatke kako bi proces bio precizniji.

NER se razvija na CoNLL (eng. Conference on Natural Language Learning) godišnjoj konferenciji koju organizira SIGNLL (eng. Special Interest Group on Natural Language Learning), specijalizirana grupa ljudi koja se bavi razvojem strojnog učenja prirodnog jezika te se svake godine izdaje novi standard za strojno učenje prirodnog jezika. Mnogi za NER koriste CoNLL2003 standard, na kojem je postignuta 90% točnost prepoznavanja entiteta. CoNLL podatkovni set je tekst razdvojen na rečenice, koje su razdovjene na riječi. Svaka riječ predstavlja jedan redak u podatkovnom setu, te je u sljedećem formatu:

RIJEČ POS CHUNK NER

gdje je „RIJEČ“ jedna riječ u tekstu, odvojena razmakom ili zarezom, „POS“ je oznaka vrste riječi u rečenici (imenica, glagol, pridjev, …), „CHUNK“ je oznaka službe riječi u rečenici (subjekt, predikat, …) a „NER“ predstavlja imenovani entitet (NER) u rečenici. „CHUNK“ i „NER“ oznake mogu sadržavati IOB (eng. Inside–outside–beginning) oznake zajedno sa svojima kako bi se povezali entiteti koji se protežu kroz više riječi, poput imena i prezimena. „B“ oznaka označava da je riječ početak entiteta, „I“ oznaka predstavlja da je riječ unutar entiteta, tj. da nije početna riječ entiteta, dok „O“ oznaka predstavlja da riječ nije dio entiteta.

Problematika u NER označavaju je u dvosmislenosti pojmova. U rečenici „Barcelona je osvojila UEFA Ligu prvaka.“, riječ „Barcelona“ je entitet organizacije, dok u rečenici „Barcelona je odabrana za grad domaćin Olimpijskih igara.“ Riječ „Barcelona“ predstavlja lokaciju.

# Python

Python je najpopularniji programski jezik za strojno učenje kreiran od strane Guido-a van Rossum-a 1991. godine. Python je spoj skriptnih i sistemskih programskih jezika te je sintaksa vrlo intuitivna, čime se programeru omogućuje fokus na problem umjesto na sintaksu. To ga također čini sporijim u izvođenju, no u današnje vrijeme, s obzirom na procesnu snagu računala, to nije problem.

Za razliku od mnogih programskih jezika koji svoje linije programskog koda odvajaju nekim jedinstvenim znakom, tipa „;“, u Python-u je jedan redak jedna linija koda. Moguće je pisati kod kroz više linija, ali se taj kod mora okružiti zagradama „()“. Također, blokovi naredbi se odvajaju uvlačenjem linije koda (određeni broj razmaka ili jedan tab) umjesto standardnih zagrada „{}“. Primjer bloka naredbi u programskom jeziku Python :

for index in range(6):

text = str(index)

print(index)

Većina funkcionalnosti potrebnih za osnovne zadatke nalazi se u osnovnom paketu biblioteka. No kako bi se ostvario puni potencijal Python-a, moguće je preuzeti i instalirati dodatne biblioteke. Dokumentacija za biblioteke može se pronaći na sljedećem linku: <https://pypi.org/project/Counter/>. Python-ova kolekcija biblioteka je jedna od najvećih i podrška zajednice je također velika. Biblioteka se može instalirati izvođenjem sljedeće naredbe:

pip install tensorflow

Python je jedan od rijetkih programskih jezika koji paralelno razvija nekoliko različitih verzija. Najnovija verzija je 3.7.1, iako se paralelno razvijaju i 2.7 i mnoge među-verzije. Objašnjenje na Python wiki stranicama: Python 2.x je zastario, Python 3.x je budućnost, no postoje neke biblioteke koje nemaju punu podršku na novijoj verziji Python-a pa je to jedan od razloga zašto postoji više verzija.

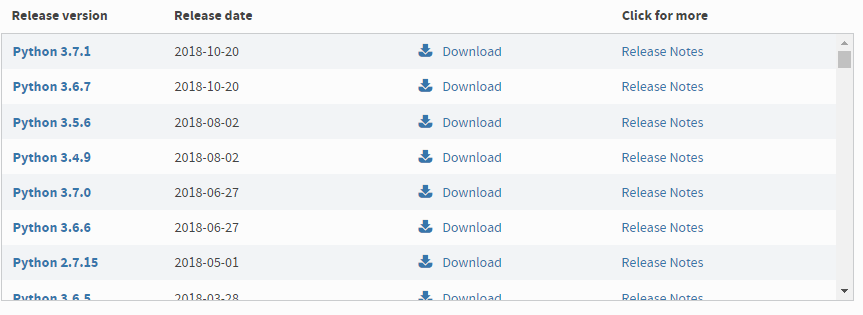
Koriste ga mnoge velike kompanije za razne svrhe, a neke od njih su:

* YouTube – velik dio serverske strane YouTube-a je pisan u Python-u
* Google – tražilica i mnoge druge aplikacije, poput Google App Engine
* Netflix – dubinska analiza podataka

## Instalacija Python-a

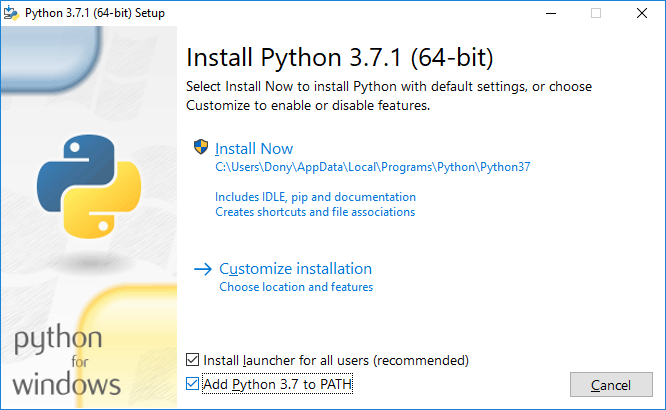


Da bismo instalirali Python na lokalno računalo, potrebno je preuzeti instalacijsku datoteku sa službene stranice, <https://www.python.org/downloads/>. Na stranici se nudi na izbor mnogo verzija Python-a, te mnogo vrsta instalacijskih paketa poput web bazirane instalacije, offline instalacije, Zip datoteke.



Slika 15 Mnogobrojan izbor verzija za preuzimanje Python programskoj jezika

Nakon preuzimanja, potrebno je pokrenuti instalacijski paket. Nakon otvaranja instalacijskog paketa, potrebno je na prozoru kliknuti „Add Python 3.7 to PATH“ okvir za izbor (eng. checkbox) kako bi se olakšalo korištenje i pokretanje Python aplikacija.



Slika 16 Python instalacijski prozor

Preostalo je još samo kliknuti „Install Now“ i pričekati da se proces završi. Kako bi verificirali instalacijski proces, potrebno je pokrenuti „Command Prompt“ (CMD). CMD se može pokrenuti pritiskom na „Windows“ tipku i slovo „R“ u isto vrijeme, čime se otvara dijalog. U dijalog je potrebno upisati „cmd“ te pritisnuti „Enter“, čime dobivamo željeni CMD. Verifikacija se može obaviti na nekoliko načina. Prvi je upisivanjem sljedeće komande:

python

čime se u CMD prozoru ispisuje tekst sljedećeg tipa:

Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)] on win32

Ako je na računalu instalirano više verzija Python-a, pri pokretanju je potrebno specificirati verziju koju želimo pokrenuti. TO je moguće sljedećom komandom:

py -3.7

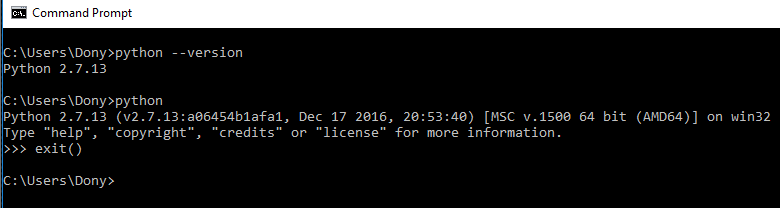
gdje je parametar „3.7“ verzija koju želimo pokrenuti. Provjeravanje verzije koja se pokreće upisivanjem komande „python“ moguće je na sljedeći način:

python –-version

nakon čega se dobije izlaz sljedećeg tipa:

Python 2.7.13

Izlazak iz komandne linije Python-a može se odraditi komandom „exit()“.



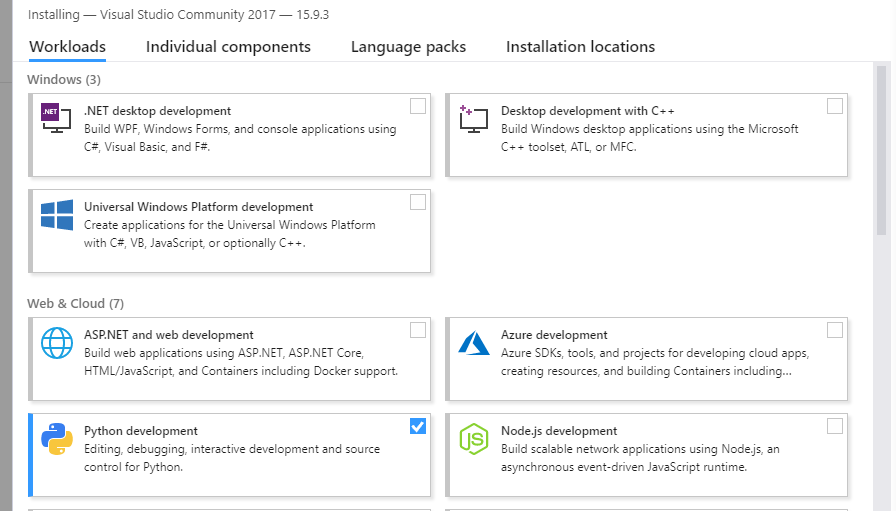
Slika 17 Python u CMD-u

# Microsoft Visual Studio

Za potrebe izrade korisničkog sučelja aplikacije, potreban je Visual Studio. *Microsoft Visual Studio* je integrirano razvojno okruženje (Engl. *integrated development environment* (IDE)) od Microsofta izdano 1997. godine. Omogućava kreiranje aplikacija i programa za operativni sustav Windows, web stranica i aplikacija, mobilnih aplikacija i još mnogo toga. Sadrži uređivač izvornog koda koji podržava *IntelliSens*[[6]](#footnote-7). *Debugger[[7]](#footnote-8)* u Visual Studio-u radi sa izvornim kodom koji je korisnik napisao kao i sa kodom iz biblioteka. Moguće je razvijati programska rješenja u nekoliko programskih jezika: C, C++, VB.NET, C# i F#. Za podršku ostalih programskih jezika mora se instalirati dodatak.

## Instalacija Microsoft Visual Studio-a

Besplatna verzija Microsoft Visual Studio-a moguće je preuzeti sa sljedećeg linka: <https://visualstudio.microsoft.com/downloads/>. Nakon preuzimanja pokretanjem instalacijske datoteke dolazi se do prozora za selekciju modula. Za ovaj projekt je potrebno odabrati modul „Python development“ kako bi se instalirala podrška za pokretanje Python koda.



Slika 18 Instalacija Visual Studio-a

# Praktični rad

Kao praktični rad razviti će se generator pitanja na temelju unesenog teksta. Tekst se unosi preko grafičkog sučelja aplikacije, koje se spaja na istrenirani model koji na tom tekstu odradi dubinsku analizu podataka za NER. U grafičkom sučelju se prikazuju pronađeni entiteti, te se za svaki pronađeni entitet generira jedno pitanje na način da se entitet zamjeni sa praznim poljem, a u odgovorima se ponude, uz točan, prikazati dva netočna odgovora iste kategorije kao i točan.

Kako bi se uopće moglo krenuti s programiranjem, potrebno je instalirati biblioteke koje se nalaze u datoteci „requirements.txt“. To se može učiniti na sljedeći način:

pip install -r requirements.txt

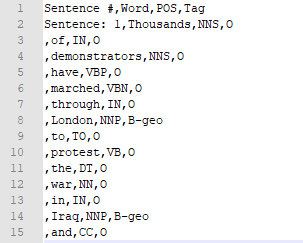
Time se instaliraju sve biblioteke potrebne za izradu praktičnog djela diplomskog rada. Potrebno je pronaći odgovarajući set podataka koji će se koristiti za treniranje modela. Zatim je potrebno kreirati model koji će se trenirati s preuzetim setom podataka. Nakon treninga je potrebno provjeriti rezultate i podesiti parametre kako bi se dobili što bolji rezultati. Kako bi se rezultati imali gdje prikazati, potrebno je kreirati korisničko sučelje, te povezati korisničko sučelje sa modelom kako bi se označeni entiteti mogli prikazati. Na kraju je potrebno dodati logiku za kreiranje pitanja.

## Pronalaženje seta podataka

Kao što je prije navedeno, gotovi setovi podataka se mogu pronaći na stranicama poput „UCI“, „Kaggle“, „Google“… Set podataka za ovaj diplomski rad preuzet je sa sljedećeg linka: <https://www.kaggle.com/abhinavwalia95/entity-annotated-corpus>. Set podataka se nalazi u CSV formatu, gdje su vrijednosti odvojene posebnim znakom, graničnikom (eng. delimiter), te prva kolona sadrži nazive kolona. CSV format je najsličniji tablicama. Preuzimanjem seta dobivaju se dvije datoteke, „ner.csv“ u kojem se podaci nalaze u sljedećem formatu:

Sentence #, Word, POS, Tag

gdje je „Sentence #“ redni broj rečenice u setu podataka, „Tag“ predstavlja NER oznaku, a ostale vrijednosti su poznate iz CoNLL formata. Druga datoteka ima mnogo opširnije podatke za svaki redak, poput prethodne i sljedeće riječi te njihovih oznaka i slično.



Slika 19 Nekoliko linija iz podatkovnog seta

Preuzeti set podataka je verificiran od strane velikog broja korisnika Kaggle servisa kao dobar, stoga dodatne verifikacije i provjere nisu potrebne. Kako bi se podaci mogli poslati modelu, potrebno ih je pretvoriti u brojčane vrijednosti.

## Pripremanje seta podataka za model

Kao reprezentaciju riječi koristit će se metoda dodavanja u listu i traženja indeksa. Za početak je potrebno učitati set podataka. Set se učitava u varijablu „data“ pozivom funkcije „pd.read\_csv()“, gdje je „pd“ Pandas biblioteka za strukturiranje i analizu podataka. Ulazni set podataka se učitava u Pandas strukturu „DataFrame“, koja radi slično kao tablica te se može pristupiti vrijednostima preko retka i preko stupca. Pošto ulazni set ima prazne vrijednosti kako se ne bi nepotrebno zapisivali nevažni podaci koji bi samo zauzimali dodatnu memoriju, potrebno je te vrijednosti popuniti. Pozivom funkcije „fillna()“ se nad setom podataka umjesto zadane vrijednost „None“ upisuju vrijednost zadane parametrom „method='ffill'“, koji uzima zadnju valjanu vrijednost te je propagira na slijedeća prazna mjesta. U slučaju na slici iznad (Slika 19), svaki će redak dobiti početnu vrijednost „Sentence 1:“.



Kod 1 Učitavanje seta te indeksiranje riječi i NER oznaka u prepareEmbeddings.py

Kako bi mogli indeksirati riječi, koje će biti ulazni podatci za model, potrebno ih je prvo izvući iz seta podataka. Kao što je prije spomenuto, vrijednosti se mogu izvlačiti preko redaka i stupaca, stoga se naredbom „data['Word'].values“ iz seta podataka „data“ izvlače sve vrijednosti „'Word'“. Kako bi vrijednosti bile jedinstvene, ta se lista riječi šalje u funkciju „get\_unique(list)“. Funkcija ima varijablu „seen“ u koju se spremaju sve vrijednosti koje su dodane u listu, te varijablu „seen\_add“ u koju se sprema „add()“ metoda. „add()“ metoda se sprema iz jednostavnog razloga. Python je dinamički jezik, odnosno pri pisanju programskog koda ne treba imenovati tipove varijabli. Kada bi se koristila obična „add()“ metoda, pri svakom dodavanju u set, Python u pozadini morao detektirati tip podatka kako bi ga mogao dodati u set, što oduzima procesnu snagu. Funkcija vraća listu jedinstvenih riječi iz ulazne liste riječi. Za potrebe modela, izračunava se i broj pronađenih riječi „n\_words“ pomoću funkcije „len()“ koja vraća duljinu nekog polja/liste. Zatim se ista stvar učini za NER oznake, koje će biti izlazni podatak iz modela.



Kod 2 Pretvaranje riječi u indekse u prepareEmbeddings.py

Generiranu listu riječi je zatim potrebno pretvoriti u indekse. Za tu svrhu se može koristiti „enumerate()“ funkcija koja prima parametar liste te iterira elemente i indekse te liste. Tako se u rječnik (eng. dictionary) mogu spremiti parovi vrijednosti „riječ : indeks“. Na taj način imamo mogućnost pretvoriti riječ u indeks i obrnuto. Također, potrebno je dodati dvije vrijednosti, jednu za nepoznate riječi „"UNK"“ te drugu za prazna mjesta „'PAD'“. Istu stvar je potrebno napraviti sa NER oznakama te sa znakovima. Riječi se dodaju na mjesto „i + 2“ kako bi se ostavilo mjesto za dvije nepoznate vrijednosti.



Kod 3 Dohvaćanje rečenica u prepareEmbeddings.py

Sljedeći korak je kreiranje rečenica uz ulaznog seta. Za tu svrhu će poslužiti prvi parametar iz podatkovnog seta „Sentence #“. Kreira se lambda funkcija, koja će vratiti listu riječi, POS oznaka i NER oznaka. „zip“ funkcija u ovom slučaju stavlja paralelno liste riječi, POS i NER oznaka te vraća po jednu vrijednost iz svake liste svakom iteracijom liste. Podatkovni set grupira se po rečenicama te mu se primjenjuje lambda funkcija. Pošto smo prilikom učitavana podatkovnog seta koristili metodu „fillna(method='ffill')“, svaka riječ u rečenici ima parametar „Sentence #“ te se prema tome grupiraju rečenice.



Kod 4 Generiranje ulaznih i izlaznih vrijednosti u prepareEmbeddings.py

Iz liste rečenica je zatim potrebno kreirati dvodimenzionalnu listu riječi, odnosno svaku riječ u rečenici je potrebno pretvoriti u indeks i dodati u listu te zatim te rečenice dodati u listu rečenica. Pošto nisu sve iste duljine, potrebno ih je svesti na istu duljinu, pomoću funkcije „pad\_sequences()“ iz „keras.preprocessing.sequence“ biblioteke. Ta funkcija prima parametar „maxlen“ koji određuje najveći broj riječi u rečenici, te će sve rečenice postaviti na tu duljinu. Kako su rečenice uglavnom manje duljine, parametar „value“ predstavlja vrijednost koja se postavlja na prazna mjesta. U gore navedenom slučaju je to „word2idx['PAD']“. Parametar „padding“ određuje da li će se prazna mjesta dodati prije („'pre'“) ili poslije („'post'“) vrijednosti iz liste, dok „truncating“ parametar određuje, ukoliko je broj riječi veći od najvećeg dopuštenog, da li se vrijednosti brišu sa početka ili sa kraja liste. Ista stvar se radi za izlazne podatke, NER oznake.

Proces pripreme ulaznih vrijednosti može trajati nekoliko minuta, stoga, kako bi se skratio proces, nakon obrade svih vrijednosti, te se liste mogu spremiti u datoteke. Koristeći biblioteku „numpy“, koja omogućuje lakše korištenje višedimenzionalnih lista, mogu se spremiti ulazne i izlazne vrijednosti u pripadajuće datoteke koristeći funkciju „save()“ koja prima dva parametra. Prvi parametar je putanja datoteke, a drugi je lista koja se želi spremiti.



Kod 5 Spremanje ulaznih i izlaznih podataka u prepareEmbeddings.py

Spremljene podatke je potrebno učitati prije cijelog procesa, te se ponovno korištenjem „numpy“ biblioteke lako mogu učitati vrijednosti u liste pomoću funkcije „load()“ koja prima parametar putanje datoteke. Prije toga je potrebno provjeriti postoje li datoteke, što omogućuje funkcija „isfile()“ iz biblioteke „os.path“. Ako datoteke ne postoje, tada se odrađuje cijeli prije navedeni postupak.



Kod 6 Učitavanje vrijednosti u liste iz datoteka u prepareEmbeddings.py

## Kreiranje modela

Za početak, potrebno je definirati parametre koji će se koristiti u modelu, poput stope učenja, broja epoha, veličine uzorka trening seta i sl.



*Kod 7 Parametri modela u model.py*

*Kod 8 Funkcija za postavljanje parametara modela u model.py*

Za početak kreira se ulazni Keras tenzor (eng. tensor). Tenzor je generalizacija vektora i matrica na potencijalno više dimenzija.[[8]](#footnote-9) Iz biblioteke „keras.models“ se poziva funkcija „Input“ kreira se ulazni tenzor za riječi. Postavlja se parametar oblika, „shape“, odnosno broj dimenzija ulaznog polja, koji je jednak maksimalnom broju riječi u rečenici. Zatim je potrebno indekse riječi normalizirati kako bi se lakše propagirali u modelu. Za to se koristi klasa „Embedding“ iz biblioteke „keras.layers“. Parametar „input\_dim“ predstavlja broj mogućih vrijednosti, što je u ovom slučaju ukupni broj riječi + dva za nepoznatu riječ i prazno mjesto, „output\_dim“ je dimenzija izlaznog vektora podataka, „input\_length“ je maksimalni broj ulaznih podataka, dok „mask\_zero“ parametar određuje ignoriranje indeksa 0 u ulaznom setu riječi. Na taj način se ignoriraju prazna mjesta u rečenicama koja su postavljena kod pripremanja seta podataka, funkcijom „pad\_sequences()“. Isto tako, dodavanjem „word2idx[‘PAD’] = 0“ je osigurano da je prazno mjesto uvijek na indeksu 0.



*Kod 9 Normalizacija znakova u model.py*

Kod je sličan kao i kod prijašnjeg koda, samo što se u „shape“ parametar postavljaju dvije vrijednosti, prva za maksimalan broj riječi u rečenici, a druga za maksimalan broj slova u riječi. Normalizacija znakova se također radi na sličan način kao i kod riječi, samo što se „Embedding“ objekt klase nalazi u omotaču (eng. wrapper) „TimeDistributed“ klase. Ta klasa omogućuje dodavanje dimenzija na sve elemente ulaznih podataka.[[9]](#footnote-10) Zatim se kreira objekt klase „LSTM“, duge kratkotrajne memorije (eng. Long Short-Term Memory) koji služi za kodiranje riječi u vektor, odnosno učenje riječi.[[10]](#footnote-11) Parametar „units“ predstavlja broj dimenzija izlaza, „return\_sequences“ predstavlja da li se uzima samo zadnja vrijednost ili cijeli set vrijednosti, te „recurrent\_dropout„ je postotak vrijednosti koje će se izbaciti.



*Kod 10 Izlazni sloj u model.py*

Sljedeći korak je spajanje tenzora normaliziranih riječi i znakova. Iz biblioteke „keras.layers“ se koristi funkcija „concatenate()“ koja spaja listu tenzora po specificiranim osima. Spojeni tenzori se zatim šalju kao parametar klasi „SpatialDropout1D“, koja će izbaciti jednu cijelu dimenziju vrijednosti svake iteracije treninga. Na taj način se sprječava navikavanje mreže na određene vrijednosti, no isto tako može doprinijeti oscilacijama u treniranju. Klasa prima jedan parametar koji određuje postotak dimenzija koje će se odbaciti. Zatim se u drugi „LSTM“ objekt predaju spojeni tenzori te se omotaju „Bidirectional“ klasom. Zbog ulaznog parametra „return\_sequences“, vraćat će samo zadnje vrijednosti, koja se proslijedi u izlazni tenzor. Izlazni tenzor je tipa „Dense“ i prima broj mogućih izlaza, što je u gornjem slučaju broj NER oznaka, te preko parametra „activation“ prima aktivacijsku funkciju, koja normalizira izlaze, tako da se dobije indeks NER oznake.



*Kod 11 Kreiranje modela i postavljane parametara u model.py*

Zatim se kreira instanca klase „Model“ iz biblioteke „keras.models“. Prvi parametar je lista ulaznih tenzora, što su u ovom slučaju tenzor riječi i tenzor znakova, te izlazni tenzor, što je u ovom slučaju tenzor NER oznaka. Modelu se zatim preko metode „compile()“ postavljaju parametri „optimizer“, „loss“ te „metrics“. „optimizer“ predstavlja optimizacijski algoritam, koji je u ovom slučaju „Adam“, koji radi vrlo dobro sa zadanim postavkama i ne treba previše konfiguriranja. „loss“ je funkcija koja predstavlja uspješnost modela, te se koristi funkcija „sparse\_categorical\_crossentropy“. Koristi se ova vrsta loss funkcije jer je NER označavanje zapravo klasifikacija, odnosno kategorizacija. „sparse“ prefiks označava da će izlazna vrijednost biti cjelobrojna (eng. integer), a ne kao kod klasične „categorical\_crossentropy“ loss funkcije koja vraća binarnu vrijednost.

## Treniranje modela



*Kod 12 Treniranje modela u model.py*

Nakon kreiranja, model se može trenirati. Treniranje modela se pokreće pozivom funkcije „fit()“ nad modelom. Funkcija prima parametre „x“, ulaz, „y“, izlaz, „batch\_size“, veličinu uzorka treninga, „epochs“, broj epoha, „validation\_split“, dio podataka koji se koriste kao validacijski podaci te „verbose“, detaljnost ispisa statusa. Kako su prije definirane dimenzije i oblici ulaznih podataka, ih je potrebno tako i predati. Kao „x“ se predaju lista indeksiranih riječi te lista indeksiranih znakova nad kojom se zove funkcija „“ iz „“ biblioteke koja mijenja oblik ulazne liste prema zadanim vrijednostima. Ista se stvar uradi i sa izlaznim vrijednostima „y“, dok se ostali parametri učitavaju iz globalnih parametara definiranih ranije u kodu. Treniranje mreže može trajati dugo, ovisno o količini podataka, veličini uzorka treninga, broju epoha i slično. Prosječno trajanje modela za prije navedenu konfiguraciju traje oko 45 minuta sa točnosti od 97.78% sa validacijskom točnošću od 97.01%.



*Kod 13 Spremanje modela i podataka o treningu u model.py*

Kako bi se kasnije model mogao koristiti za NER označavanje, potrebno ga je spremiti. Model se sprema preko funkcije „save()“ koja prima parametar lokacije na koju se sprema model. Isto tako, kako bi se mogao vizualizirati postupak treniranja, potrebno je spremiti izvještaj treniranja uz pomoć „pandas“ biblioteke, pozivom funkcije „to\_csv()“. Nakon treniranja, model se može testirati.

## Testiranje modela

Testiranje modela može se odraditi na više načina. Jedan od načina je da se testira sa podacima iz ulaznog seta podataka, tako da se ulazni set podjeli na trening set i testni set. Razdvajanje je lako ostvarivo pomoću funkcije „train\_test\_split()“ iz biblioteke „sklearn.model\_selection“. Funkcija prima dvije liste podataka, te parametre „test\_size“ koji određuje koliki je udio testnog seta u ukupnom podatkovnom setu i „random\_state“, koji predstavlja broj koji će se koristiti u nasumičnom dijeljenu podatkovnog seta. Funkcija vraća podijeljene ulazne setove. Funkcija prima više ulaznih setova, no moraju biti jednakih dimenzija, stoga su „X\_word“ i „y“ poslani zajedno, a „X\_char“ posebno.



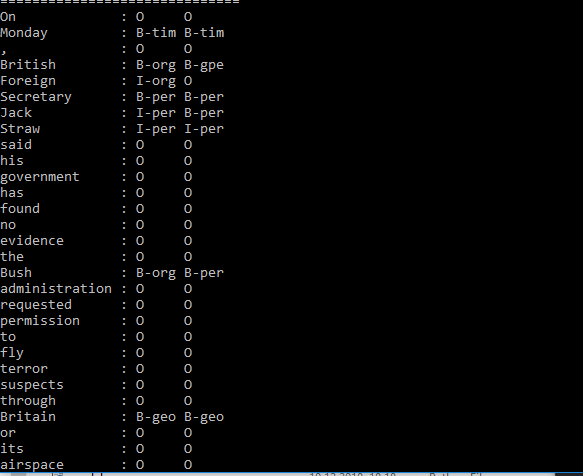
*Kod 14 Testiranje modela u main.py*

Nakon podjele setova, poziva se „train()“ funkcija iz biblioteke „model“ koja je ranije kreirana za potrebe praktičnog djela diplomskog rada. „train()“ prima prije podijeljene setove za treniranje, te parametre koji su pripremljeni ranije. Nakon treniranja, poziva se funkcija „predict()“, koja prima testne dijelove setova, te rječnike za pretvaranje riječi u indekse te putanju do spremljenog modela. Prije testiranja potrebno je učitati model pomoću funkcije „load\_model()“ iz biblioteke „keras.models“. Funkciji se pošalje parametar putanje modela, te se u varijablu spremi učitani model. Pozivom funkcije „predict()“ nad učitanim modelom testira se model. U ovom slučaju se šalju samo ulazni parametri, dok je izlazni parametar povratna vrijednost funkcije. Ako je model vratio ikakve vrijednosti, uzima se vrijednost sa nasumičnog indeksa te se ispisuju vrijednosti sa tog indeksa.



*Kod 15 Testiranje modela u model.py*

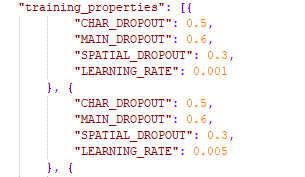
Kako bi dobili lijepo formatirani izlaz, u funkciji za ispis „print()“ definiramo broj mjesta koje će ta varijabla zauzimati. Tako za riječi imamo 15 mjesta, za točnu NER oznaku 5 mjesta, a za predviđenu NER oznaku ne formatiramo jer nije potrebno. Kako se ne bi ispisivala prazna mjesta koja su dodana prilikom pripremanja podataka, provjerava se uvjet „if word != 0:“, jer smo na početku odredili da prazno mjesto ima indeks 0.



*Slika 20 Primjer testa modela*

## Mijenjanje parametara

Na slici (Slika 20) može se vidjeti da, iako je točnost modela 97.8%, i dalje imamo relativno netočne podatke. Stoga je potrebno promijeniti parametre i pratiti promjene. Stoga je kreirana datoteka koja sadrži nekoliko konfiguracija modela. Mijenjaju se stopa učenja, koja utječe koliko brzo algoritam uči, te stopa izbacivanja vrijednosti na različitim tenzorima, koji mogu imati velik utjecaj na brzinu i točnost učenja, jer sprječavaju navikavanje mreže.



*Slika 21 Konfiguracije modela*

Za početak provjerimo da li postoji trenirani model za svaku konfiguraciju u konfiguracijskoj datoteci. Funkcija „try\_get\_graph\_files()“ provjerava u određenom direktoriju da li postoji izvještajna datoteka o treniraju, te ukoliko ne postoji, diže grešku (eng *error*), tipa „FileNotFoundError“ koja javlja da jedna od datoteka nije pronađena te da je potrebno ponoviti postupak treniranja. No ako sve datoteke postoje, u varijablu „graph\_files“ se spremaju sve datoteke koje su generirane tijekom prijašnjeg procesa treniranja.



*Kod 16 Provjera postajanja izvještaja treniranja u iterate.py*

Treniranje počinje učitavanjem parametara konfiguracije iz konfiguracijske datoteke. Zatim se podjeli ulazni set podataka na trening i test setove, te se generira deskriptivno ime modela, kako bi poslije jednostavnije odredili koju konfiguraciju koristi određeni model. Zatim se modelu postavljaju parametri pomoću funkcije „set\_parametars()“ te se pokreće trening modela. Kompletna logika se nalazi u petlji, te se vrti onoliko puta koliko ima konfiguracija u konfiguracijskoj datoteci, a to je 21. Nakon završetka treninga, u varijablu „graph\_files“ se spremaju novonastali izvještaji o treniranju.

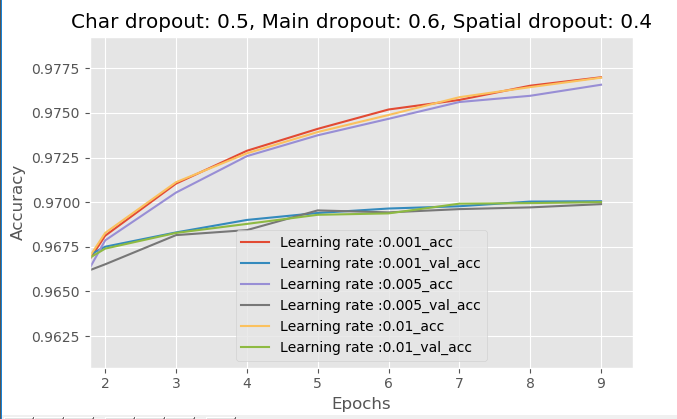


*Kod 17 Iterativno treniranje u iterate.py*

### Rezultati mijenjanja parametara

Nakon završetka treniranja, dobiju se 42 datoteke u mapi „iterationData“, od kojih su pola izvještaju, a ostala polovica modeli. Izvještaji se mogu prikazati na grafovima segmentiranim po jednom parametru koji se mijenja u konfiguraciji, te za svaki parametar koji mijenjamo postoje 3 konfiguracije sa različitim stopama učenja.

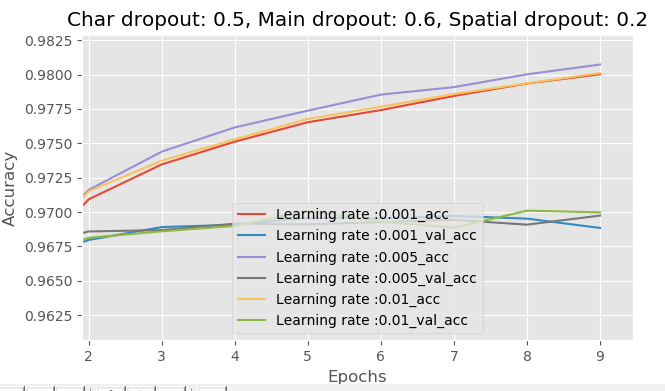
Ukupna prosječna točnost je 97.76%, čime je prosjek lošiji od prije istreniranog modela za 0.02%, dok je ukupna prosječna validacijska točnost 96.99%, što je manje od prijašnje vrijednosti za 0.04%. Te su razlike zanemarive, ako uzmemo u obzir da su sve konfiguracije ostvarile točnost veću ili jednaku 97.65%.



*Slika 22 Najlošija konfiguracija modela*

Najlošija konfiguracija ima prosječnu stopu točnosti od 97.65%, što je 0.11% ispod prosjeka, dok je stopa točnosti validacijskog seta podataka 97%, što je više od prosječne točnosti za 0.01%. Stopa učenja nema prevelik utjecaj na konačni rezultat, osim što se tijekom cijelog grafa može vidjeti da srednja stopa učenja u početku ima najsporiji rast, što se propagira kroz cijeli graf te na kraju zaostaje nekoliko promila u točnosti. Na grafu se vidi da su vrijednosti vrlo slične, te da graf ima još prostora za rast, što dovodi do zaključka da je broj epoha od 10 definitivno premalen. Povećanjem parametra „SPATIAL\_DROPOUT“ za 0.1 dobivamo najlošije rezultate. To je parametar koji određuje odbacivanje jedne cijele dimenzije ulaznih tenzora riječi i znakova, čime se zaključuje da bi smanjivanje parametra moglo donijeti bolje rezultate.

Najbolja konfiguracija ima prosječnu stopu točnosti od 98.02% čime je za 0.26% bolja od prosječne vrijednosti, te validacijsku stopu točnosti od 97.96%, što je 0,03% niže od prosječne validacijske točnosti. U



*Slika 23 Najbolja konfiguracija modela*

# Zaključak

Područje strojnog učenja je vrlo zanimljivo i još je uvijek u razvoju. Ne možemo ni zamisliti što se sve može napraviti sa strojnim učenjem, koliko se stvari može automatizirati. Tek se u zadnjih nekoliko godina tržište počelo zanimati za proizvode sa umjetnom inteligencijom i dan danas taj interes samo raste. Google i Microsoft ulažu ogromne resurse u umjetnu inteligenciju jer su uvidjeli potencijal tog područja.

Dubinskom analizom podataka mogu se iz naočigled nasumičnih i nebitnih podataka izvući informacije ili iz hrpe podataka izvući neka informacija. U primjeru gore je pokazano kako je iz hrpe recenzija moguće izvući je li opće mišljenje dobro ili loše o određenoj marki automobila. Također je moguće, kasnije, pretražiti što se korisnicima ne sviđa te tako promijeniti te stvari u budućim verzijama automobila.

# Literatura

**ONLINE**

**Wikipedia**

* <https://hr.wikipedia.org/wiki/Strojevi> (30.6.2018.)
* <https://hr.wikipedia.org/wiki/Inteligencija> (30.6.2018.)
* <https://hr.wikipedia.org/wiki/Umjetna_inteligencija> (30.6.2018.)
* <https://en.wikipedia.org/wiki/Machine_learning> (1.7.2018.)
* <https://en.wikipedia.org/wiki/Bayesian_network> (1.7.2018.)
* <https://en.wikipedia.org/wiki/Directed_acyclic_graph> (2.7.2018.)
* <https://en.wikipedia.org/wiki/Amazon_(company)> (8.12.2018)

**Keras**

* <https://keras.io/> (8.12.2018.)

**Quintagroup**

* <https://quintagroup.com/cms/python/google> (8.12.2018.)

**Amazon**

* <https://aws.amazon.com/machine-learning/> (8.12.2018.)

**Azure**

* <https://azure.microsoft.com/> (8.12.2018.)

**Google**

* <https://cloud.google.com/> (8.12.2018.)

**CoNLL**

* <http://www.conll.org/2018> (9.12.2018.)

**TensorFlow**

* <https://www.tensorflow.org/guide/tensors> (10.12.2018.)

**Knjige**

* Data Mining, Third Edition, Ian H. Witten, Eibe Frank, Mark A. Hall, 2011.
* Data Mining: Technologies, Techniques, Tools, and Trends, Bhavani Thuraisingham, 1999.
* Programming Python, Mark Lutz, O'Reilly, 2006.
* Named Entities: Recognition, Classification, and Use, Satoshi Sekine, Elisabete Ranchhod, 2009.

**Akademski članci**

* Open Source Data Mining Programs: A Case Study on R, Muhammet Sinan Başarslan, Fatih Kayaalp, Travanj 2018.
* Introduction to Information Retrieval - Stanford NLP, Online edition, 2009

# Prilozi

Izvorni kod: <https://github.com/kvrabec/NerTagger>

**TEHNIČKO VELEUČILIŠTE U ZAGREBU**

**POLITEHNIČKI SPECIJALISTIČKI DIPLOMSKI STRUČNI STUDIJ**

**Specijalizacija elektrotehnika**

Kristijan Vrabec

JMBAG: 0246051246

Dubinska analiza podataka u svrhu generiranja kviz pitanja

DIPLOMSKI RAD br. I 969

Povjerenstvo:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Zagreb, prosinca 2018.

1. NLTK – Natural Language Toolkit – biblioteka u programskom jeziku Python za obradu teksta [↑](#footnote-ref-2)
2. HTML - HyperText Markup Language – prezentacijski jezik za izradu web stranica [↑](#footnote-ref-3)
3. Dijakritički znakovi - znakovi sa preglasima, kvačicama i sl. [↑](#footnote-ref-4)
4. Stopword - riječi koje nemaju utjecaj na značenje teksta [↑](#footnote-ref-5)
5. SQL - Structured Query Language – strukturirani jezik za upite, koristi se u bazama podataka [↑](#footnote-ref-6)
6. Microsoftova implementacija automatskog predlaganja nastavka koda (engl. *auto complete*) [↑](#footnote-ref-7)
7. Program koji pomaže u traženju i rješavanju grešaka u kodu [↑](#footnote-ref-8)
8. Tensors - <https://www.tensorflow.org/guide/tensors> [↑](#footnote-ref-9)
9. TimeDistributed - <https://www.tensorflow.org/api_docs/python/tf/keras/layers/TimeDistributed> [↑](#footnote-ref-10)
10. LSTM - <https://keras.io/layers/recurrent/#lstm> [↑](#footnote-ref-11)