UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

JEZIKOVNO MODELIRANJE Z REKURZIVNIMI NEVRONSKIMI MREŽAMI

DAVID VAC

2020

ZAKLJUČNA NALOGA

VAC

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Jezikovno modeliranje z rekurzivnimi nevronskimi mrežami

(Language modeling with recurrent neural networks)

Ime in priimek: David Vac
Študijski program: Računalništvo in informatika
Mentor: izr. prof. Jernej Vičič
Somentor: doc. dr. Branko Kavšek

Koper, september 2020

Ključna dokumentacijska informacija

Ime in PRIIMEK: David VAC

Naslov zaključne naloge: Jezikovno modeliranje z rekurzivnimi nevronskimi mrežami

Kraj: Koper

Leto: 2020

Število listov: 28 Število slik: 5

Število referenc: 28

Mentor: izr. prof. Jernej Vičič

Somentor: doc. dr. Branko Kavšek

Ključne besede: Jezikovno modeliranje, rekurzivne nevronske mreže, pomnenje, sekvenca, treniranje

Izvleček:
Zaključna projektna naloga predstavlja implementacijo jezikovnega modeliranja z uporabo rekurzivnih nevronskih mrež. Najprej bralcu predstavimo področje jezikovnega modeliranja. Temu pa sledi opis kako delujejo rekurzivne nevronske mreže v povezavi z jezikovnim modeliranjem. Nato bralcu predstavimo samo implementacijo programa in kako poteka učenje samega programa skozi več ponovitev. Na koncu naloge so pa še predstavljeni rezultati testov izbranih parametrov.

Key document information

Name and SURNAME: David VAC

Title of final project paper: Language modeling with recurrent neural networks

Place: Koper

Year: 2020

Number of pages: 28 Number of figures: 5

Number of references: 28

Mentor: Assoc. Prof. Jernej Vičič, PhD

Co-Mentor: Assist. Prof. Branko Kavšek, PhD

Keywords: Language modeling, recurrent neural networks, memory, sequence, training

Abstract: The final project presents the implementation of language modeling using recursive neural networks. We first introduce the reader to the field of language modeling. This is followed by a description of how recursive neural networks work in conjunction with language modeling. Then we present to the reader only the implementation of the program and how the learning of the program itself takes place through several repetitions. At the end of the thesis, the results of tests of selected parameters are presented.

Zahvala

Zahvalil bi se mentorju izr. prof. Jerneju Vičiču in somentorju doc. dr. Branku Kavšku za vso podporo in usmeritve tako pri zaključnem delu, kot v času izobraževanja. Prav tako bi se zahvalil družini in prijateljem za vso podporo, ki so mi jo izkazali na izobraževalni poti.
Hvala!

Kazalo vsebine

Kazalo slik in grafikonov

Seznam kratic

RNN	Reccurent neural network (Rekurzivna nevronska mreža)
LSTM	Long short term memory (Dolg kratkoročni spomin)
RESNET	Residual neural network (Preostanek nevronske mreže)

1 Uvod

Jezikovno modeliranje je uporaba statističnih in probabilističnih tehnik za določanje verjetnosti dane sekvence znakov, besed ali stavkov v danem kontekstu. Osnovna ideja tega procesa je da uporabimo statistične podatke prej podanih znakov in na podlagi njih predvidimo naslednji znak. Problem nastane pri tem ko si nemoremo zapomniti dovolj podatkov, ki nam jih poda nek želen korpus in tako nemoremo natančno napovedati nadaljevanje sekvence. Tukaj pa vstopi koncept rekurzivnih nevronskih mrež, ki nam z svojo sposobnostjo pomnenja pomaga pri bolj natančnemu napovedovanju. In sicer rekurzivna nevronska mreža si zapomne vhodne podatke z uporabo enostavne zanke tako da vzame trenutne vhodne informacije in jih doda naslednjemu vhodu. S tem načinom pomnimo relacije med besedami in tako lahko treniramo naš jezikovni model skozi dolge sekvence podatkov in na koncu z pomočjo uteži ki jih prejme vsaka tako imenovana relacija med besedami napovemo napovedovanje besedila. Lepota uporabe nevronskih mrež je v tem da jih lahko uporabimo za različne tipe vhodnih in izhodnih podatkov kot so na primer prevajanje jezikov, sentimentalno in slikovno analizo in celo prepoznavanje govora, kjer rekurzivna nevronska mreža napoveduje nadaljevanje fonetičnih segmentov.

Zaključna naloga je razdeljena na teoretični in praktični del. V prvem delu bralca seznanimo z osnovnimi koncepti jezikonvega modeliranja in delovanjem rekurzivnih nevronskih mrež. Nato pa mu predstavimo kako jezikovno modeliranje deluje v praksi na točno določenem primeru.

2 Jezikovno modeliranje

Jezikovni modeli analizirajo besedilne podatke tako da zagotovijo podlago za nadaljno napoved. Največ se uporablja za obdelavo naravnega jezika, zlasti za take tehnike, ki ustvarjajo besedilo kot izhod, kot so na primer samodejno prevajanje ali odgovarjanje na vprašanja. Jezikovni modeli determinirajo verjetnost znaka, besede ali stavka glede na analizo besedilnih podatkov. Le tej interpretirajo te podatke tako da jih spustijo skozi nek alogritem ki določa pravila za kontekst v naravnem jeziku. Nato model uporabi ta pravila pri jezikovnih nalogah za natančno napovedovanje ali ustvarjanje novih stavkov. Jezikovni model se v bistvu nauči lastnosti osnovnega jezika in jih nato uporabi za razumevanje in predikcijo novih stavkov. [1]

2.1 Vrste jezikovnega modeliranja

Strokovna literatura loči več verjetnostnih pristopov k modeliranju jezika, ki se razlikujejo glede na namen le tega. S tehnične perspektive se različne vrste razlikujejo glede na količno besedilnih podatkov, ki jih analizirajo in glede na algoritem oziroma matematiko ki jo uporabijo za analizo. Na primer jezikovni model za samodejno ustvarjanje stavkov uporablja drugačno analizo in matematiko kot jezikovni model, ki je zasnovan za iskalne poizvedbe.

2.1.1 N-gram

Ta je najbolj pogost in relativno enostaven pristop k jezikovnemu modeliranju. Ustvari se porazdelitev verjetnosti za zaporedje n, kateri je lahko poljubno število in določa velikost n-grama ali zaporedja besed, ki jim je verjetnost izračunana iz učnega korpusa. Na primer, če je n enak 5 model dodeli verjetnosti sekvencam velikosti n. V bistvu je n količina konteksta, ki ga model mora upoštevati.Ideja o uporabi n-gram modela je mogoče zaslediti nazaj do eksperimenta v teoriji informacij z strani Clauda Shannona, ki je postavil vprašanje kakšna je verjetnost pojavitve nekega znaka na podlagi prej podanega niza le-teh. [2]

Unigram Je najbolj enostaven tip n-gram modela, ki pri svojem pristopu ne upošteva nobenega konteksta in vsako besedo ali izraz oceni neodvisno. Unigram modeli običajno

obravnavajo naloge obdelave jezika, kot je iskanje informacij. Unigram je temelj bolj specifične različice modela, imenovanega model verjetnosti poizvedbe, ki uporablja iskanje informacij za pregled zbirke dokumentov in primerja najprimernejšega s točno določeno poizvedbo.

2.1.2 Dvosmerni jezikovni model

Je model, ki za razliko od n-gramskih modelov, ki analizirajo besedilo v eno smer, dvosmerni modeli analizirajo besedilo v obe smeri, nazaj in naprej. Ti modlei lahko z uporabo druge besede v besedilu predvidijo katero koli besedo v stavku ali besedilu. Dvosmerno pregeldovanje poveča natančnost rezultatov. Ta vrsta jezikovnega modeliranja se pogosto uporablja v aplikacijah strojnega učenja in govora. Google na primer uporablja dvosmerni model za obdelavo iskalnih poizvedb. [3]

2.1.3 Eksponentni jezikovni model

Je poznan tudi kot model z največjo entropijo in je kompleksnejsa vrsta jezikovnega modeliranja od n-gramov. Model ocenjuje besedilo z enačbo, ki združuje funkcijske funkcije in n-grame. V bistvu ta vrsta določa lastnosti in parametre želenih rezultatov, za razliko od n-gramov pa parametre analize pušča dvoumnejše, na primer ne določa posameznih velikosti gramov. Model temelji na načelu entropije, ki pravi da je porazdelitev verjetnosti z največ entropije najboljša izbira. Eksponentni modeli so zasnovani tako, da maksimizirajo navzkrižno entropijo, kar zmanjša količino statistilnih predpostavk, ki jih je mogoče dati. To uporabnikom omogoča boljše zaupanje rezultatom teh modelov.

2.1.4 Jezikovno modeliranje z nevronskimi mrežami

Modeliranje s pomočjo nevronskih mrež uporabljajo nenehne predstavitve ali vdelave besed za svoje napovedi. Ponavljajoče prostorske vgradnje pomagajo ublažiti tako imenovano »prekletstvo dimenzij v jezikovnem modeliranjue«, saj se jezikovni modeli trenirajo na vse večjih in večjih besedilih. Število možnih zaporedij se eksponentno povečuje z velikostjo besedišča, kar povzroča težavo z redkostjo podatkov in zato je potrebna statistika za pravilno ocenjeno verjetnost. Nevronske mreže se tej težavi izognejo tako, da besede predstavljajo porazdeljeno kot nelinearne kombinacije uteži v nevronski mreži. [4]

2.2 Pomen jezikovnega modeliranja

Jezikovno modeliranje je v sodobnih aplikacijah NLP ključnega pomena. To je razlog, da stroji lahko razumejo kakovostne informacije. Vsaka vrsta jezikovnega modela pretvori kvalitetne informacije v kvantitativne informacije. To ljudem omogoča, da v omejenem obsegu komunicirajo s stroji, tako kot med seboj. Uporablja se neposredno v veliko različnih panogah, vključno s tehnologijo, financami, zdravstvom, prometom. Poleg tega je verjetno večina ljudi, ki berejo v nekem dnevu na nek način stopila v stik z jezikovnim modelom, naj bo to iskanje z Googlom, funcijo samodejnega dokončanja besedila ali sodelovanjem z galsovnim asistentom. Same korenine jezikovnega modeliranja segajo daleč nazaj v leto 1948, ko je Claude Shannon objavil članek z naslovom »Matematična teorija komunikacije«. V njej je podrobno opisal uporabo stohastičnega modela, imenovanega veriga Markov, za izdelavo statističnega modela za zaporedje črk v angleškem besedilu. Ta članek je močno vplival na telekomunikacijo, postavil je temelje informacijske teorije in jezikovnega modeliranja. Markov model se uporablja še dan danes, n-grami pa so natančno vezani na koncept jezikovnega modeliranja.

2.3 Primeri uporabe jezikovnega modeliranja

2.3.1 Prepoznavanje govora

Jezikovno modeliranje je temelj naraven obdelave jezika. Uporablja se v različnih panogah obdelave jezika kot so, prepoznavanje govora. Interdisciplinirana podkategorija računalništva in računalniške lingvistike, ki razvija metodologije in tehnologije za prepoznavanje in prevajanje govorjenega jezika v besedilo s pomočjo računalnika. Nekatere tehnike prepoznavanja govora zahtevajo učenje, kjer posameznik bere besedilo ali izoliran besednjak v sistem. Nato sistem analizira posameznikov glas in ga uporablja za natančno nastavitev govora te osebe, kar pripelje do večje natančnosti. Na drugi strani pa poznamo še tehnike, ki ne potrebujejo učenja in so neodvisne od sistema, imenujemo jih »neodvisne od govornika«. Prepoznvanje govora je proti koncu dvajsetega stoletja naredilo velik korak naprej in našla se je široka paleta uporabe, vse od računalniških igrač, nadzoru različnih inštrumentov do narekovanja. Siri je eden najvidnejših primerov govornega vmesnika, ki prikazuje vpliv prepoznavanja govora v današnji družbi. Možno je še veliko več aplikacij kot so na primer notranje vgrajeni sistemi za avte, kjer voznik namesto uporabe konzole pritisne gumb, ki omogoča prepoznavanje zvoka z katerim lahko uporabnik kliče, izbira radijske postaje, predvaja glasbo in še marsikaj. Prepoznavanje govora je naredilo tudi velik napredek v zdravstvu, kjer lahko uporabniki tega sistema diktirajo želeno besedilo in nato ustvarijo zdravstven dokument katerega

je nato samo še potrebno pregledati in podpisati. Razpon prepoznavanja govora sega tudi do aplikacij v vojski, kjer bi lahko upravljaji visoko tehnološka vozila, vse od letal do vozil na tleh, kar bi zelo razbremenilo porabo vojaških sredstev in posledično tudi manj žrtev na bojnih ozemljih. Seveda pa nesmemo pozabiti na aplikacije tega pristopa jezikovnega modeliranja v telefonski industrij, izobraževalnih panogah, kjer je prepoznvanje govora izredno uporabno pri učenju drugih jezikov, tekočem govoru in govornih sposobnostih. Nenazadnje pa imajo velik korist z napredkom prepoznavanjem govora še ljudje z posebnimi potrebami, kjer bi lahko posamezniki z dizleksijo kljub temu da katere besede nemorejo prav izgovoriti, bi programska oprema zaznala pravo besedo. [5]

2.3.2 Strojno prevajanje

Jezikovno modeliranje se uporablja tudi v strojnem prevodu, ki raziskuje uporabo programske opreme za prevajanje besedila ali govora iz enega jezika v drugega. Na osnovni ravni strojni prevod izvaja mehansko nadomeščanje besed v enem ali drugem jeziku z besedami drugega, vendar se le redko ustvari dober prevod, saj je potrebno prepoznavanje celotnih stavkov in njihovih najbljižjih sosedov v ciljnem jeziku. Ta problem se rešuje s korpusnimi statističnimi in nevronskimi tehnikami, ki vodijo v boljši prevod, obravnavanju razlikm prevajanju idiomov in izolacije anomalij. Obstajajo trije glavni pristopi k strojnem prevajanju. Prvi pristop je strojno prevajanje, ki temelji na pravilih. Slednji uporablja prenos, medjezično strojno prevajanje in paradigme strojnega prevajanja na osnovi slovarja. Ta vrsta prevajanja se večinoma uporablja pri ustvarjanju slovarjev in slovničnih programov. V nasprotju z drugima dvema pristopoma RBMT vključuje več informacij o jezikoslovju izvornega in ciljenga jezika, z uporabo morfoloških in skladenjih pravil ter semantične analize obeh jezikov. Drugi izmed treh pristopov je statistični strojni prevod, ki poskuša ustvariti prevode s pomočjo statističnih metod, ki temeljijo na dvojezičnih besedilnih korpusih, kot so kanadski korpus Hansard, alngleško-francoski zapis kanadskega parlamenta in zapis ecropskega parlamenta. Kadar so takšni korpusi na voljo je mogoče pri prevajanju podobnih besedil doseči dobre rezultate, vendar so takšni korpusi še vedno redki za številne jezikovne pare. Nazadnje pa obstaja še pristop k strojnem učenju z uporabo nevronske mreže, ki uporablja učenje programske opreme, kako doseže najboljši rezultat. Ta proces porabi velike količine procesne moči, zato se pogosto izvaja na gradičnih enotah procesorjev. Tak pristop, ki temelji na globokem učenju je v zadnjih letih naredil velik skok naprej in sam Google je oznanil, da njihove prevajalske storitve dandanes uporabljajo slednji pristop namesto statističnega pristopa. [6]

2.3.3 Slovnično označevanje

V korpusni lingvistiki je označevanje dela govora ali tudi drugače imenovano slovnično označevanje, postopek označevanja besede v korpusu, ki ustreza določenemu delu govora in temelji na njegovi definiciji in kontekstu. Poenostavljeno obliko tega postopoka se uči šolo obvezne otroke in sicer pri določanju besed kot so samostalnik, glagol, pridevnik in podobno. Nekoč izvajano na roko, se dan danes slovnično označevanje izvaja v kontekstu računalniškega jezikoslovja z uporabo algoritmov, ki povezujejo diskretne izraze in skrite dele govora z naborom opisnih oznak. Označevanje dela govora delimo na dve značilni skupini in sicer označevanje na osnovi pravil in stohastično označevanje. Ena najstarejših tehnik označevanja dela govora je označevanja na osnovi pravil. Označevalci na osnovi pravil delujejo na dvostopenjski arhitekturi. Prva stopnja uporabi slovar ali leksikon opisnih oznak za označevanje posamezne besede in ji tako dodeli seznam možnih delov govora. Nato pa sledi druga stopnja, ki se izvede, če ima beseda več možnih oznak. V tem primeru se uporabi velike sezname ročno napisanih pravil o razlikovanju, ki vsaki besedi dodelijo točno določeno oznako glede na njeno definicijo in kontekst . Razlikovanje se pa lahko izvede tudi z analizo jezikovnih značilnosti besede skupaj z njenimi predhodimi in naslednjimi besedami, na primer če je prejšnja beseda členek mora biti naslednja beseda samostalnik. Druga tehnika označevanja je stohastično slovnično označevanje. Ta tehnika temelji na verjetnostnem pristopu. Vsaki besedi z pomočjo vadbenega korpusa dodelimo svojo oznako, vendar moramo paziti pri izbiri korpusa, saj če slednji ne vsebuje neke beseda le ta nima svoje dodeljene oznake. Najprepostejši stohastični pristop je besedni frekvenčni pristop, pri katerem označevalci razlikujejo besede na podlagi verjetnosti, da se beseda pojavi z določeno oznako. [7]

2.3.4 Razčlenjevanje

Razčlenjevanje ali sintaktična analiza je postopek analiziranja niza simbolov bodisi v naravnem jeziku, računalniškij jezikih ali podatkovnih strukturah, skladnih z slovničnimi pravili. V računalniškem jezikoslovju se izraz uporablja za računalniško analizo stavka ali drugega niza besed, razdeljene v manjše elemente. Posledično se ustvari drevo razčlenitve, ki prikazuje skladen odnos med samimi elementi. Drevo samo pa lahko vsebuje tudi semantične in druge informacije. Razčlenjevanje se tako uporablja v nekaterih sistemih strojnega prevajanja in obdealve naravnega jezika. Naravni jezik ni lahko razčleniti, saj je v strukturi človeškega jezika mogoče najti dvoumnosti katere računalniški programi ne morejo napovedati. Zato je potrebno pred razčlenjevanjem določiti pravila, ki bodo pomagala pri samem postopku. Za izvajanje sintaktične analize potrebujemo programsko komponento imenovano razčlenjevalec, ki sprejme vhodne

podatke in iz njih zgradi podatkovno strukturo, najbolj pogosto je to drevo razčlenitve, s katerim predstavimo strukturo vnosa medtem ko preverjamo pravilno skladnjo. Naloga razčlenjevalca je v bistvu ugotoviti ali in kako vhod izhaja iz začetnega simbola slovenice. To je mogoče na dva načina, razčlenjevanje od zgoraj navzdol ali pa od spodaj navzgor. Razčlenjevanje od zgoraj navzdol je mogoče razumeti kot poskus iskanja najbolj levih izpeljav odsekov vhodenga toka z iskanjem drevesa razčlenitve danih formalnih pravil slovnice. Razčlenjevanje od spodaj navzgor pa deluje tako, da razčlenjevalnik lahko začne z vhodom in ga poskuša napisati kot začetni simbol. Intuitivno razčlenjevalec poskuša najti najosnovnejše elementem nato elemente, ki vsebujejo te najosnovnejše elemente in tako naprej. [8]

2.3.5 Prepoznavanje rokopisa

Prepoznavanje rokopisa je tudi ena izmed tehnik, ki uporablja jezikovno modeliranje in sicer uporablja zmnožnost računalnika za sprejem in razlago razumljivega ročno napisanega vnosa iz virov, kot so papirni dokumetni, fotografije, zasloni na dotik in druge naprave. Sistem za prepoznavanje rokopisa tako obravnava, izvede pravilno segmentacijo in najde najbolje verjetne besede, ki se ujemajo z pisavo rokopisa. Poznamo dve tehniki prepoznavanja rokopisa in sicer »off-line« in »on-line« metodo. Princip »off-line« prepoznavanja rokopisa vključuje samodejno pretvorbo besedila v sliki v črkovne kode, ki jih je mogoče uporabiti v računalniških aplikacijah in aplikacijah za obdelavo besedila. Podatki dobljeni s to obliko se štejejo za statičen prikaz rokopisa. Prepoznavanje rokopisa »off-line« je sorazmerno težavno, saj imajo različni ljudje različne sloge rokopisa. »off-line« delimo še na tradicionalne in moderne metode. Tradicionalne metode začnejo najprej z eksrakcijo znakov, kar pomeni da moramo najprej skenirati obrazec ali dokument za kasnejšo ekstrakcijo. Sledi prepoznavanje posameznih znakov. Slednji postopek izvedemo z uporabo mehanizma za prepoznavanje ustreznega računalniškega znaka. Nazadnje pa še uporabimo ekstrakcijo funckij, ki omogoča mehanizmu za prepoznavanje večji nadzor nad lastnostmi uporabljenimi pri identifikaciji skeniranih znakov. Za razliko od tradicionalih metod, ki bolj poudarjajo segmentiranje posameznih znakov za prepoznavanje rokopisa, se sodobne metode osredotočajo na prepoznavanje vseh znakov v segmenitrani vrstici besedila. Poudarjajo zlasti tehnike strojnega učenja, ki imajo zmožnost učenja vizualnih lastnosti pri čemer se izognejo omejujočim lastnostim, kot so različni stili in pisave vsakega posameznika. »on-line« prepoznavanje rokopisa vključuje samodejno pretvrobo besedila, saj je napisano na posebnem digitalizatorju ali dlančniku, kjer senzor analizira premike konice pisala. Dobljeni signal pa se pretvori v črkovne kode, ki jih lahko uporabimo v računalniških aplikacijah in aplikacijah za obdelavo besedila. Proces »on-line« prepoznavanja rokopisa delimo na

tri korake, predobdelavo, ekstrsakcijo funkcij in razvrstitev. Namen predobdelave je zavreči nepomembne informacije v vhodnih podatkih, ki lahko negativno vplivajo na prepoznavanje. Predobdelava običajno obsega binarizacijo, normalizacijo, vzorčenje, glajenje in ekstrakcijo uporabnih informacij. Drugi korak je ekstrakcija funckij, ki iz dvodimenzionalnega ali večdimenzionalnega vektorskega polja, prejetega iz algoritmov predhodne obdelave, pridobi podatke. Namen tega koraka je izpostaviti pomembne indormacije za model prepoznavanja. Ti podatki vključujejo inforamcije, kot so pritisk peresam hitrost ali spremembe strani pisanja. Zadnji korak »on-line« prepoznavanje rokopisa je klasifikacija, V tem koraku se uporabljajo modeli za preslikavo prej izvlečenih funkcij v različne razrede in s tem identifikacijo znakov ali besede, ki jih funkcije predstavljajo. [9]

2.3.6 Iskanje informacij

Iskanje informacij je dejavnost pridobivanja virov informacijskega sistema, ki so pomembni za potrebo po informacijah iz zbirke teh virov. Iskanja lahko temeljijo na polnem besedilu ali drugem indeksiranju na podlagi vsebine. Postopek iskanja informacij se začne, ko uporabnik vnese poizvedbo v sistem. Poizvedbe so formalne izjave o potrebah po informacijah, na primer iskalni nizi v spletnih iskalnikih. Pri iskanju informacij poizvedba ne identificira enotnega predmeta v zbirki. Namesto tega se lahko več predmetov ujema z poizvedbo, morda z različnimi stopnjami ustreznosti. Predmet je entiteta, ki jo predstavljajo informacije v zbirki vsebine ali baze podatkov. Poizvedbe uporabnikov se primerjajo z podatki iz baze. Toda v nasprotju s klasičnimi poizvedbami SQL baze podatkov se pri iskanju informacij podatki lahko ujemajo ali pa ne z dano poizvedbo. Večina sistemov iskanja informacij izračuna številčno oceno glede na to, kako dobro se vsak predmet v bazi podatkov ujema s poizvedbo in razvrsti predmete v skladu s to vrednostjo. Nato se uporabnikom prikažejo najvišje uvrščeni predmeti. Če uporabnik ni zadovoljen z rezultati poizvedbe lahko slednjo izpopolni in postopek razvrščanja se ponovi. Glede na aplikacijo so lahko podatkovni objekti na primer, besedilni dokumenti, slike ali video posnetki. Pogosto dokumenti sami niso shranjeni neposredno v sistemu iskanja inforamcij, ampak so sistemu predstavljeni kot nadomestki dokumentov ali metapodatki. [10]

3 Rekurzivne nevronske mreže

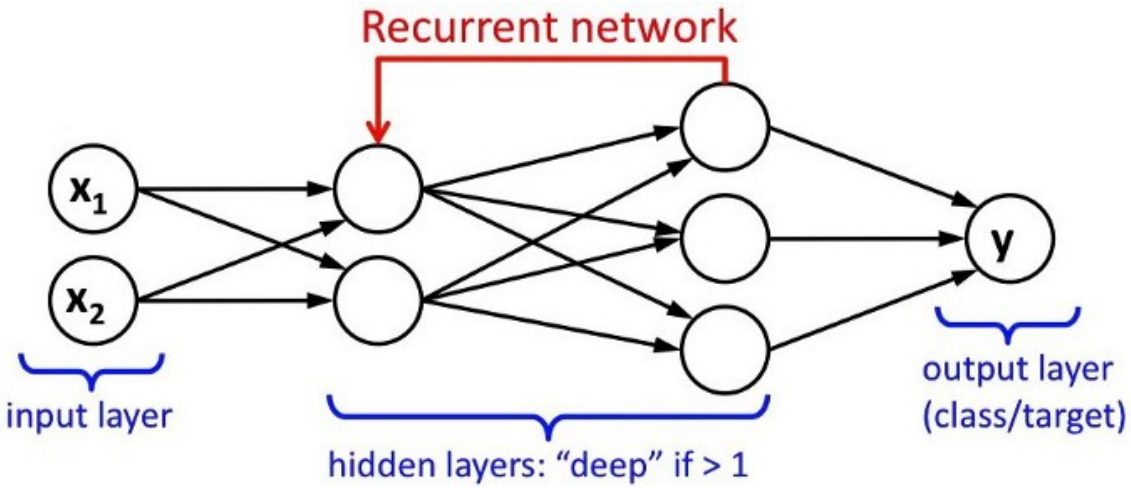
3.1 Nevronska mreža

Nevronska mreža je vrsta algoritmov, ki si prizadevajo prepoznati osnovne odnose v naboru podatkov s postopkom, ki posnema način delovanja človeških možganov. Podatkovne strukture in funkcionalnost nevronskih mrež so zasnovane tako, da simulirajo asociativni spomin. Nevronske mreže se učijo s predelavo primerov, od katerih vsak vsebuje znan vhod in rezultat, ki skupaj tvorita verjetnostno prilagojene povezave med obema in so shranjene v sami podatkovni strukturi mreže. Tako je učenje nevronske mreže iz danega primera različno glede na stanje pred in po obdelavi primera. Po podajanju zadostnega števila primerov mreža postane sposobna predvideti rezultate iz vložkov z uporabo asociacij, zgrajenih iz vzorčnega niza in če je nevronski mreži zagotovljena zanka povratnih informacij o točnosti svojih napovedi se poveza še naprej izboljšujejo in posledično imamo vedno večjo raven natančnosti. Torej obstaja veliko razmerje med številom in raznolikostjo primerov, ki jih obdela nevronska mreža in natančnostjo njenih napovedi. Ker so pa nevronske mreže neločljivo povezane z oblikovanjem asociacij, lahko le te tvorijo nepričakovane asociacije in razkrijejo odnose in odvisnosti, ki prej niso bile znane. Prva, ki sta uporabila koncept nevronskih mrež sta Warren McCulloh in Walter Pitts. Leta 1943 sta ustvarila računalniški model za nevronske mreže, ki temelji na algoritmih imenovanih pragovna logika. Ta model je utrdil pot raziskovanja nevrnoskih mrež na dva pristopa. En pristop je bil osredotočen na biološke procese, drugi pa na uporabo nevronskih mrež za umetno inteligenco. [11]

3.1.1 RNN

Rekurzivna nevronska mreža spada v razred umetnih nevronskih mrež, kjer povezave med vozlišči tvorijo usmerjen graf po časovnem zaporedju. Izhajajoč iz naprej podajajočih nevronoskih mrež, RNN-ji lahko uporabljajo svoj pomnilnik oziroma notranje stanje za obdelavo zaporedij vhodov s spremenljivo dolžino. Zaradi tega so uporabne za naloge, kot so nesegmentirano, povezano prepoznavanje rokopisa ali prepoznavanje govora. RNN je vrsta nevronske mreže pri kateri se izhodi iz prejšnjega stanja prenesejo naprej in uporabijo kot vhod za trenutno stanje. V tradicionalnih nevronskih omrežjih se vsi vhodi in izhodi neodvisni drug od drugega, toda v primerih, ko je potrebno

predvideti naslednjo besedno zvezo, je potrebno znanje o predhodnih besedah in zato si je treba zapomniti prejšnja stanja. Problem pomnenja predhodnih besednih zvez reši glavna in najpomembnejša značilnost RNN-ja imenovana skrito stanje, ki pomni informacije o prejšnjih zaporedjih.



Slika 1: Struktura rekurzivnih nevronskih mrež [28]

RNN imajo pomnilnik, ki si zapomni vse informacije o tem kaj je že bilo izračunano. Za vsak vhod uporablja iste parametre, saj izvja enako naloga na vseh vhodih ali skritih plasteh za izdelavo izhoda. To zmanjša zapletenost parametrov, za razliko od drugih nevroniskih mrež. [12]

Predpostavimo, da obstaja globje omrežje z enim vhodnim slojem, tremi skritimi stanji in enim izhodnim slojem. Kot pri vsaki nevronski mreži, bo imel vsak skriti sloj imel svoj nabor uteži in pristranskosti. Recimo skrito stanje 1 ima utež in pristranskosti (w1, b1), (w2, b2) za drugi skriti stanjej in (w3, b3) za tretjo skrito stanje. To pomeni, da so plasti med seboj neodvisne, čemur sledi, da si prejšnjih izhodov ne zapomnijo. Sedaj pa RNN naredi naslednje. RNN pretvori neodvisna stanja v odvisna stanja z zagotovljanjem enakih uteži in pristranskosti vsem slojem, s čimer se zmanjša kompleksnost povečevanja parametrov in zapomni se vsak prejšnji izhod, tako da se vsak izhod poda kot vhod v naslednji skriti sloj. Zato je mogoče te tri plasti združiti tako, da so uteži in pristranskosti vseh skritih plasti enake.

3.2 Arhitektura

3.2.1 Polno ponavljajoč

Polno ponavljajoči RNN so mreže, nevronom podobnih vozlišč, organizirana v zaporedne plasti. Vsako vozlišče je povezano z usmerjeno povezavo do vsakega drugega

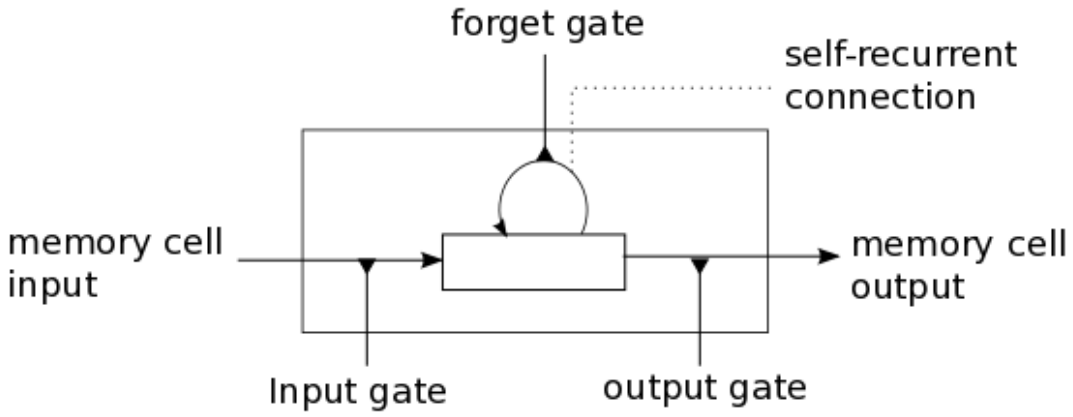
v naslednjem zaporednem sloju. Vsako vozlišče ima časovno spreminjajočo se aktivacijo v realni vrednosti in vsaka povezava ima spremenljivo realno utež. Vozlišča so bodisi vhodna, sprejemajo informacije zunaj omrežja, izhodna vozlišča, prinašajo rezultate ali skrita vozlišča, ki sprejemajo informacije na poti od vhoda do izhodnega vozlišča. Poznamo dve motodi učenja in sicer nadzorovano učenje in okrepljeno učenje. Nadzorovano učenje je naloga učenja funkcije, ki preslika vhod v izhod na podlagi vhodno-izhodnih parov. Funkcijo sklepa na podlagi označenih podatkov o učenju, sestavljenih iz niza primerov iz prej naučenih informacij. Pri nadzorovanem učenju je vsak primer para, sestavljen iz vhodnega elementa, ponavadi vektorja in želene izhodne vrednosti, imenovane tudi nadzorni signal. Nadzorovan algoritem učenja analizira podatek od vadbi in izdela sklepno funckijo, ki jo lahko uporabimo za preslikavo novih primerov. Okrepljeno učenje pa se ukvarja s tem, kako bi morali programski agenti ukrepati v nekem danem okolju, da bi maksimirali pojem kumulativne nagrade, torej da izberejo najprimernejši rezlutat. Okrepljeno učenje se od nadzorovanega razlikuje po tem, da ne potrebuje predstavitve označenih vhodno-izhodnih parov. Namesto tega je poudarek na isaknju ravnotežja med raziskovanjem novih informacij in izkoriščanjem trenutnega že naučenega znanja. [13]

3.2.2 Rekurziven

Rekurzivna nevronska mreža je ustvarjena z uporabo istega niza uteži rekurzivno preko diferencirane grafovske strukture s prečkanjem strukture v topološkem vrstnem redu. Takšna omrežja so navadno usposobljena z obratnim načinom samodejnega razlikovanja. Lahko obdeluje porazdeljene predstavitve struktur, kot so logični izrazi. Poseben primer rekurzivnih nevronski mrež je RNN, čigar struktura ustreza linearni verigi in so bile uporabljen pri procesiranju naravnega jezika. [14]

3.2.3 LSTM

Dolgoročni kratkoročni pomnilnik je sistem globokega učenja, ki preprečuje težave z izginjajočim gradientom. LSTM preprečuje, da bi povratne napaka izginile ali eksplodirale, Namesto tega lahko napake tečejo nazaj skozi neomejeno število virtualnih slojev, razgrnjenih v prostoru. To pomeni, da se lahko LSTM nauči naloge, ki zahtevajo spomine na dogodkem, ki so se zgodili na tisoče ali celo miljone diskretnih časovnih korakov nazaj. LSTM lahko deluje celo z daljšimi zamudami med pomembnimi dogodki in zna upravljati s signali, ki mešajo nizko in visokorekvenčne komponente. [15]



Slika 2: Struktura LSTM oblike rekurzivnih nevronskih mrež. [29]

3.3 Učenje

3.3.1 Problem izginjajočega gradienta

Učenje rekurzivnih nevronskih mrež lahko privede do težav z izginjajočim gradientom, ko treniramo le te z metodami bazinarimi na učenju z gradientom in algoritmom povratnega širjenja. Pri takšnih metodah vsaka utež nevronske mreže prejme posodobitev, sorazmerno z delnim izvodom funkcije napake glede na trenutno težo v vsaki ponovitvi učenja. Težava je v tem, da je v nekaterih primerih gradient izredno majhen, kar prepreči gradientu da bi posodobil svojo vrednost teže in tako v najslabšem primeru morda celo popolnoma ustavil nevronsko mrežo pri nadaljnem učenju in posodabljanju. Za premagovanje težave z izginjajočim gradientom je bilo predlaganih več metod. Ena od njih je večstopenjska hierarhija omrežij Jürgena Schmidhuberja, ki je predhodno izpolnjevala eno stopnjo z nenadzorovanim učenjem in natančno prilagojenim povratnim širjenjem. Tu se vsaka stopnja nauči stisnjene predstavitve, ki je nato podana naslednji stopnji. [16] Ena izmed rešitev je že prej omenjeni LSTM, ki z svojo sposobnostjo globokega učenja prepreči zmanjševanje gradienta do te mere, da bi slednji preprečil nadaljno učenje rekurzivne nevronske mreže. Napredek v strojni opremi je prav tako pripomogel k zmanjšanju težav z izginjajočim gradientom, saj je v zadnjih 20 letih moč računalnika povečala za približno miljonkrat, kar je pomenilo, da je algoritem povratenga širjenja izvedljiv kar nekaj plasti globlje preden je prepoznana težava z izginjajočim gradientom. Nazadnje pa še eden najnovejših in najučinkovitejših pristopov, k rešitvi težav z izginjajočim gradientom, je pristop z tako imenovano preostalo nevronsko mrežo. Pred tako imenovanim ResNetom je bilo ugotovljeno, da bi

globje omrežje lahko imelo višjo napako pri učenju kot plitvo omrežje. To intuitivno lahko razumemo tako, da bolj ko je omrežje globoko več podatkov izgubimo, čemur sledi, da je izhod iz plitvih slojev postajal vedno manj pomemben kar pomeni slabši končni rezultat. Zato je bil ResNet sestavljen iz sorazmerno plitvih mrež, ki dejansko ne odpravljajo težav z izginjajočim gradientom, ampak ohranijo gradientni tok skozi celotno globino omrežja in se z sestavljanjem številnih sklopov plitvih omrežij izognejo teževam z izginjajočim gradientom. [17]

3.3.2 Problem eskplodirajočega gradienta

Na drugi strani pa moramo med učenjem RNN paziti še na eksplodirajoči gradient. Težava z eksplodirajočim gradientom se pojavi, ko se naberejo večje količine gradientnih napak, kar povzroči zelo velike posodobitve uteži nevronskih mrež med treningom. Rezultat velikih posodobitev je nestabilno nevronsko omrežje in v skrajnem primeru lahko pride celo do preliva vrednosti uteži, čigar posledica je nezmožnost nadaljnega učenja mreže. Obstaja veliko različnih pristopov za reševanje težav z eksplodirajočim gradientom. V globokih nevronskih omrežjih lahko eksplodirajoče gradiente rešimo z preoblikovanjem same omrežja tako, da ima manj slojev. Tako se podobno, kot pri pristopu ResNet, izognemo eksplodirajočemi gradientu. V RNN lahko pride do eksplozije gradienta zaradi prirojene nestabilnosti pri usposabljanju take vrste omrežja, ki ga povzroči povratno širjenje, ki v bistvu spremeni RNN v globko večplastno nevronsko omrežje. V takih vrstah omrežij ta problem rešujemo z uporabo LSTM, ki z svojo sposbnostjo dolgoročnega učenja prepreči samo eksplozijo gradienta in preliv vrednosti uteži. Težave z eksplodirajočim gradientom se lahko še vedno pojavljajo v zelo globokih večplastnih omrežjih v primeru velike velikosti serij in LSTM-ja z zelo velikimi dolžinami zaporedja vhodov. V takem primeru med učenjem preverjamo omrežje in če vidimo da prihaja do eksplozij gradienta lahko slednjemu omejimo velikost. Taki metodi pravimo gradientno rezanje. Konkretno se vrednosti gradientnih napak preverja glede na vrednost praga in se vrednost gradienta obreže ali nastavi na neko mejno vrednost. [18]

3.4 Uporaba RNN

RNN so zelo močni modeli strojnega učenja in uporablja se jih na najrazličnejših področjih zaradi možnosti uporabe RNN-jevih zaporednih podatkov. Uporablja se jih pogosto na področjih, kot so jezikovno modeliranje in ustvarjanje besedila, strojni prevod, prepoznavanje govora, ustavarjanje opisov slik, video označevanje, povzetek besedil, analiza klicev, zaznavanje obrazov, prepoznavanje slik.

Jezikovno modeliranje in ustvarjanje teksta Zmogljivost obravnave vnosa spremenljive dolžine naredi omrežja RNN izredno primerna za modeliranje zaporednih podatkov, kot so stavki v naravnem jeziku. To je kot nalašč za jezikovno modeliranje, saj je naravni jezik zaporedje besed in vsaka naslednja besed aje odvisna od besed, k pridejo in so pred njo.

Strojno učenje RNN se uporablja tudi za prevajanje besedila iz enega jezika v drugega. Skoraj vsi prevajlski sistemi, ki se uporabljajo danes, uporabljajo RNN. Trenutno ena najbolj priljubljenih aplikacij za strojno preverjanje Google Translate uporablja model RNN-ja. Obstajajo še številne druge po meri narejene aplikacije RNN-ja, ki se uporabljajo za natančnejše in bolj omejeno iskanje vsebin na različnih platformah, kot so Ecommerce, Flipkart, Amazon in Ebay. [19]

Prepoznavanje govora Prepoznavanje govora je tudi eno izmed področjih, kjer so omrežja RNN pogosto uporabljena. Predvsem jih uporabljajo za napovedovanje fonetičnih segmentov, pri čemer so zvočni valovi iz medija predstavljajo vhodni vir. Tako se vhodni podatki najprej obdealajo in prepoznajo s strani nevronske mreže. Rezultat je raznolika zbirka vhodnih zvočnih valov. Informacije vsebovane v zvočnem valu so nato dadatno razvrščene po nameri in s pomočjo govora, povezane s poizvedbo. Sledi proces razvrščanja zvočnih valov v fonetične segmente in jih z uporabo RNN aplikacije sestavimo v kohezivne besede. Rezultat je sestavljen iz vzorca fonetičnih segmentov, ki so logično sestavljeni v edinstveno celoto. [20]

Generiranje opisa slike RNN se uporablja za opis kaj se dogaja znotraj slike tako da uporabi segmentirane podatke z strani konvolucijske nevronske mreže in ponovno sestavi opis.

Označevanje videa Podobno kot pri generiranju opisa slik z uporabo RNN uporabimo segmentirane podatke številnih okvirjev videa in sestavimo označevalce videa.

Povzemanje besedila RNN aplikacije lahko pomagajo tudi pri povzemanju vsebin različnih literatur in prilagajanju le-teh, predvsem so uporabne za zelo dolga besedila, saj RNN lahko obdeluje besedila poljubne dolžine. Na primer, če želi založnik prikazati povzetek ene od svojih knjig na zadnji strani, da bi bralcem pomagal spoznati vsebino literature, ki je v njej, je RNN metoda izredno koristna. [21]

Analiza klicnih centrov To lahko štejemo za najbolj pomembnih aplikacij RNN-jev na področju obdelave zvoka. Analiza klicev podjetjem ponuja informacije zakaj je

podporno osebje uspelo pomagati klicateljem oziroma zakaj niso morali pomagati in kakšni so bili koraki pri reševanju klicateljevih problemov. To znanje je nato mogoče preučiti in uporabiti za druge podobne scenarije ali pa za usposabljanje novih kadrov. Celoten postopk analize je mogoče avtomatizirati z pomočjo RNN-jev, ki samo obdela in sintetizira dejanski pogovor med klicateljem in delavcem. Tak sintetiziran govor je mogoče nadalje vzeti kot vhod v algoritem za merjenje čustev in občutkov, povezanih z različnimi deli pogovra, kar bi podjetjem pomagali zvedeti ali so bile stranke zadovoljne s storitvijo in podporo. Čeprav je to mogoče storiti z enostavnim opazovanjem ljudi bi avtomatizacija podjetjem pomagala kvantificirati podatke in iz njih pridobiti vpogled za v prihodnje, saj so informacije o analizah shranjene v bazi podatkov.

4 Implementacija

4.1 Motivacija

Tipkovnice so dan danes velik del našega vsakdanjega življenja, saj jih praktično uporabljamo v vsakem računalniškem okolju s katerim stopimo v stik. Podobno pomoč uporabljajo tudi oametni telefoni, vendar je struktura predvidevanja nadaljevanja besedila bazirana na drevesni strukturi in ne uporablja verjetnosti za predvidevanje katera beseda bi lahko nadaljevala dano sekvenco. Da bi zmanjšali naš trud pri tipkanju večine tipkovnic, ki jih uporabljamo omogočajo možnost naprednega napovedovanje. Kar pomeni, da nam po vsakem pritisku tipkovnica poda najverjetnejšjo predikcijo na podlagi prej uporabljenih znakov. Tako se nam ustvarja lokalni slovar besed in fraz, ki jih pogosto uporabljamo. Ob vsaki uporabi neke nove besede ali fraze bo naša tipkovnica slednjo oštevilčila in ob vsaki novi uporabi dodala število uporab k temu oštevilčenju.
Na začetku so te predikcije lahko smešno slabe, vendar se bodo z časom in treningom tudi te izboljšale. Vendar to ni dovolj. Saj če tipkovnico uporabljamo vsakodnevno za pogovore z prijatelji nam slednja ne bo služila pri pisanju diplomskih ali seminarskih nalog. Tukaj vstopijo modeli jezikovnega modeliranja z uporabo rekurzivne nevronske mreže. Z možnostjo obdelave poljubno dolgega teksta, ki ga imajo RNN-ji lahko vstavimo katerokoli želeno besedilo, vse od shakespeareja do biologije človeškega telesa. Po kratkem učenju tipkovnice nam bo le-ta omogočila bolše predikcije in manj časa porabljenega pri iskanju želenih izrazov ali fraz. S čimer bomo hitreje in lažje opravili z zadanimi nalogami.

4.2 Definicija problema

V tem poglavju bo definirana implementacija našega jezikovnega modela z uporabo rekurzivnih nevronskih mrež. Problem napovednih zmogljivosti, ki jih uporabljajo trenutne tipkovnice je, da delujejo tako, da podajajo predikcije glede na utež prej podanih znakov, ki so že bili uporabljeni. Torej, če neka beseda še ni bila uporabljena, ji napovedna zmogljivost ni mogla še določiti uteži, kar pomeni da želene besede med predikcijami nebomo našli. Vendar mi želimo, da je besedam iz nekega korpusa, že

ob samem začetku pisanja nekega besedila ali dokumenta določena utež, kar rešimo z uporabo rekurzivnih nevronskih mrež.Cilj uporabe rekurzivne nevronske mreže je izkoristiti možnost pomnenja prejšnjih stanj in obdelavo korpusa neomejene dolžine. Tako bi uporabniku zagotovili kontekstualno pravilne predikcije in s tem olajšali pisanje želenih dokumentov in odpravo pravopisnih napak.
Rekurzivne nevronske mreže omogočajo pregled ne le ternutnega vhoda, ampak tudi tistega, ki je bil dan korak nazaj. Kar pomeni, da odločitev, sprejeta v prejšnjem časovnem koraku neposredno vpliva na odločitev v trenutnem koraku. RNN definirajo ponovitve v časovnih korakih, ki ga poda enačba:

$$S_t = f(S_{t-1} * W_{rec} + X_t * W_x)$$

Slika 3: Enačba ponovitev v časovnih korakih

Kjer St stanje v časovnem koraku t, Xt je vhod v času t, Wrec in Wx pa sta parametra uteži. RNN tako lahko izračunajo trenutno stanje St iz trenutnega vhoda Xt in iz prejšnjega stanja St-1 ali napovejo naslednje sanje St+1 iz trenutnega stanja St in trenutnega vhoda Xt. Konkretno modelu podamo sekvenco znakov in ga vprašali za predikcijo naslednjega. V naslednjem koraku dodamo novo naučeno stanje in tako naprej dokler ne pridobimo celotno besedo. Dve glavni težavi mučita rekurzivne nevronske mreže in sicer problem izginjajočega in eksplodirajočega gradienta. V tradicionalnih RNN-jih lahko gradientni signal velikokrat pomnožimo z maso teže. Tako lahko vrednost uteži prehodne matrike igra pomembno vlogo. Če so uteži v matriki majhne, postane gradientni signal na vsakem vadbenem koraku manjši, s čimer učenje postane zelo počasno ali se morda celo popolnoma ustavi. Temu pravimo problem izginjajočega gradienta. Nasportno pa se problem eksplodirajočega gradienta nanaša na to, da so uteži v prehodni matriki tako velike, da lahko povzročijo prelivanje in s tem izbuljanje naučenih informacij. Za izogib tem problemom uporabimo posebno vrsto RNN, LSTM, ki se uči dolgoročnih obveznosti. Uvede novo strukturo imenovano spominska celica, ki je sestavljena iz štirih elementov, vhodnega podatka, pozabljivih in izhodnih vrat, ter nevrona, ki ohranja rekurzivno povezavo. LSTM se spopadajo z izginjajočim gradientom, tako, da ohranjajo napako, ki jo je mogoče povratno širiti skozi čas in sloje. Z ohranjanjem bolj konstantne napake omogoča učenje dolgoročnih odvisnosti. Na drugi strani pa se eksplodirajoči gradient nadzira z omejevanjem gradienta, kar pomeni, da gradientu določimo neko vrednost katere ne sme preseči.

4.3 Programska oprema

Implementacijo programa začnemo z namestitvijo vseh potrebnih knjižnic za delovanje rekurzivnih nevronskih mrež. Za implementacij programa smo izbrali okolje python. Python je splošni, visoko ravenski programski jezik, ki ga je leta 1991 ustvaril Guido van Rossum. Pythonova oblikovna filozofija poudarja berljivost kode z vidno uporabo pomembnega prostora. Njegovi jezikovni konstrukti in objektno usmerjen pristop ciljata k temu, da pomagata programerjem pri pisanju jasne in logične kod za majhne in velike projekte. [22] Python podpira več programskih paradigm, vključno s strukturnim, objektno usmerjenim in funkcionalnim programiranjem. Nmaesto, da bi v jedro programskega jezika vgradili vso funkcionalnost, je python zelo razširljiv. Zaradi kompaktne modularnosti je postal še posebej priljubljen kot sredstvo za dodajanje programljivih vmesnikov obstoječim aplikacijam. Van Rossumova vizija majhnega jedrnega jezika z veliko standardno knjižnico in enostavno razteznim tolmačem je izvirala iz njegovih frustracij z ABC, ki je zagovarjal nasproten pristop. [23]
Ob tem uporabili še Tensorflow platformo za strojno učenje, ki ima celovit, fleksibilen ekosistem orodij, knjižnic in virov kateri raziskovalcem omogočajo uporabo najnovejših tehnologij na področju strojnega učenja. Temu pa smo dodali še odprtokodno knjižnico nevrnoskih mrež napisano v jeziku Python imenovano Keras. Zasnovan je tako, da omogoča hitro eksperimaentiranje z globokimi nevronskimi omrežji in se osredotoča na uporabniku prijazno, modularno in razširljivo okolje. Razvit je bil v okvirju raziskovalnega dela projetka ONEIROS (odprti nevro-elektronski inteligentni robotski operacijski sistem), njegov glavni avtor in vzdrževalec pa je François Chollet, Googlov inženir. Keras ni postavljen kot samostojen okvir za strojno učenje, ampak je zasnovan kot vmesnik, ki ponuja bolj intuitiven nabor abstrakcij za olajšanje razvoja modelov globoke učenja ne glede na zaledje računalniškega zaledja. [24]
Za učenje omrežja smo uporabili korpus Friedricha Nietzcheja Beyond Good and Evil. Ta tekst ni prevelik zato se bo naš model relativno hitro učil. Kar nam pride zelo prav, saj želimo, da je naš program čim bolj natančen, to pa pomeni, da bomo potrebovali čim več iteracij učenja programa v želji doseči naš cilj, ki pa je največja možna natančnost podanih predikcij. Pred začetkom obdelave korpusa moramo slednjega še tako rekoč očistiti vseh nepotrebnih znakov. Začnemo z pretvorbo besedila v majhne črke tako, da program prebere celoten korpus in ga preoblikuje. Nato pa se znebimo še vseh posebnih znakov, kot so pike, vejice, klicaji in podobno, saj slednjih ne želimo med končnimi rezultati. Po pretvorbi besedila z uporabo inzeniringa lasnosti sestavimo seznam unikatnih besed, s katerimi nato ustvarimo slovar, ki vsebuje besede kot ključe in njihov položaj v slovarju kot vrednost. Lasnostni inzeniring je postopek uporabe domenskega zanja za pridobivanje lasnosti iz neobdealnih podatkov s pomočjo tehnik

podatkovnega rudarjenja. Sama lasnost je atribut, ki jo delijo vse neodvisne enote na katerih je treba opraviti analizo ali napovedovanje. Tako je lahko vsak atribut lasnost, če je le uporabljena v modelu učenja. Te novo pridobljene informacije je mogoče uporabiti za izboljšanje zmogljivosti algoritmov strojnega učenja. [25]
Sledi kreacija dveh polj, ki ju uporabimo za shranjevanje prejšnjih besed in naslednje besede. Tako z uporabo for zanke zapolnimo prvo polje z želeno dolžino sekvence prešnjih besed, v našem primeru je to 5 besed in drugo polje z besedo, ki sledi podani sekvenci. Ta beseda je pa tudi ta, ki jo želimo predvideti.
V naslednjem koraku kreiramo dve novi polji, tako imenovani numpy polji za shranjevanje funkcij in pripadajočih oznak z uporabo prej kreiranih polj z sekvenco prejšnjih besed in naslednje besede. Numpy je knjižnica programskega jezika python, ki ponuja podporo velikim večdimenzionalnim nizom in matrikam, skupaj z veliko zbirko matematičnih funckij visoke ravni za delovanje na teh nizih. [26] Slednjo strukturo ustvarimo, ker strojno učenje ne prepozna besed. Zato moramo le te pretvorit v enakovredne numerične vrednosti, da lahko lažje predelamo podatke. Tako ob ujemanju besed uporabimo postopek imenovan »one-hot« kodiranje z katerim ustvarimo vektorje za vsako ujemajočo unikatno besedo in sicer, da njeno pozicijo nastavimo na ena in vse ostale na nič ter zagotovimo da strojno učenje ne dela razlik med besedami.Po končanem kodiranju pridobimo matriko najdenih vektorjev, ki jo kasneje uporabimo za predikcije. [27]
Po končanem zadnjem koraku preprocesiranja je naš program pripravljen na grajenje jezikovnega modela. Za kreacijo našega jezikovne modela smo uporabili eno slojni LSTM model z 128 nevroni, popolnoma povezan sloj, ki omogoča da je vsak vhod odvisen od prejšnjega izhoda in softmax funkcijo tudi znano kot softargmax, ki vzame vhodni vektor in ga normalizira v verjetnostno porazdelitev, katero kasneje uporabimo za podajanje predikcij. Po uporabi funkcije softmax bo vsak normaliziran vektor imel vrednost med 0 in 1, skupna vsota pa ne bo presegla 1.
Procesu grajenja jezikovnega modela pa sledi treniranje. Naš model bomo trenirali na 1,2,5,10,20 in 50 epohih. Epoh je vrednost koliko krat bodo vektorji posodobili svoje uteži, torej več kot bo epohov bolj natančne bodo uteži. Slednji proces pa bomo opravili z RMS optimizatorjem, ki z uporabo premikajočega povprečja kvadratnih gradientov normalizira sam gradient. Tako uravnavamo velikost korakov med gradienta in sicer v primeru visokega gradienta z zmanjšanjem koraka preprečimo eksplozijo gradienta in izgubo naučenih informacij. V primeru nizkega gradienta pa korak povečamo in se tako izognemo problemu izginjajočega gradienta, ki bi nam lahko popolnoma ustavil proces učenja modela. Po vsakem uspešnem učenju shranimo naš model, saj smo porabili kar nekaj časa za samo učenje in ga nato lahko po želji tudi naložimo nazaj. Vsako uspešno učenje modela poda tudi rezultate evaluacije programa, ki nam na koncu pomagajo pri

sami analizi uspešnosti treniranja jezikovnega modela in nam pomaga odgovoriti na vprašanje ali smo usepli doseči naš cilj čim bolj natančnih in kontekstualno pravilnih predikcij.

Jezikovni model smo pripravili na učenje korpusa, ga zgradili in natrenirali. Vse kar nam še ostane je proces predvidevanja novih besed z uporabo tega modela. Najprej moramo pripravi tekst tako, da vhodno sekvenco spremenimo v funkcijski vektor. Nato napišemo vzorčno funkcijo z katero naš model napove željeno število najverjetnejših besed. Nazadnje pa še vse skupaj povežemo skupaj in z pomočjo priprave vhoda in vzorčne funkcije model napove najverjetnejše besede z uporabo seznama le teh.

5 Evalvacija

5.1 Opis testov

Za namen testiranja delovanja jezikovnega modela sem opravil 6 različnih poskusov. Cilj je bil oceniti po kolikih iteracijah učenja bo model začel podajati kontekstualno pravilne predikcije. V prvem testu smo modelu dali nalogo, da le enkrat obdela korpus, v drugem testu model korpus obdela dva krat in tako naprej do zadnjega šestega testiranja, kjer pa je model petdeset krat obdelal dan korpus. Seveda lahko pričakujemo, da bo prvi test pokazal najslabše rezultate, saj gre model le enkrat skozi celotno besedilo. Prav tako lahko pričakujemo, da ne bomo imeli težav z izginjajočim ali pa eksplodirajočim gradientom, saj smo točno za ta namen uporabili LSTM rekurzivno nevronsko mrežo, ki odpravlja teževe z slednjima gradientoma.
Testiranje je potekalo tako, da smo jezikovnemu modelu podali korpus Friedricha Nietzcheja Beyond Good and Evil v obdelavo. Nato smo morali modelu še podati poljuben stavek iz katerega je jezikovni model izluščil sekvenco dolgo 5 besed in model je bil pripravljen na učenje. Kot rezultat model na koncu poda seznam možnih besed za nadaljevanje prej izbrane sekvence in ob vsaki uspešni obdelavi korpusa pa model poda še analizo podatkov v obliki natančnosti le teh in koliko inforamcij je bilo izgubljeno tekom učenja.
Vsa opravljena testiranja so v obdelavo dobila enak korpus in enak poljuben stavek iz katerega se je vedno izluščila enaka sekvenca besed.

Test 1 Model enkrat obdela besedilo.
Izguba podatkov: 6.5598
Natančnost: 0.1000
Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['be', 'say', 'the', 'have', 'make']

Test 2 Model dvakrat obdela besedilo.
Izguba podatkov: 6.0822
Natančnost: 0.1398

Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['be', 'say', 'the', 'have', 'make']

Test 3 Model petkrat obdela besedilo.
Izguba podatkov: 5.4552
Natančnost: 0.2681
Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['be', 'that', 'as', 'have', 'the']

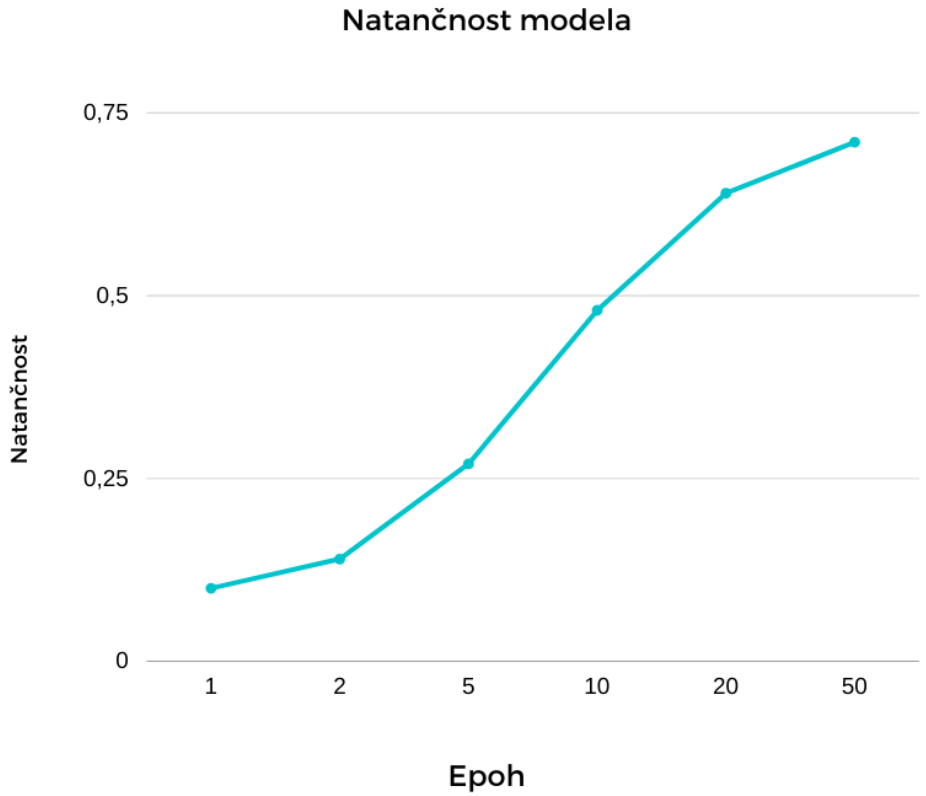
Test 4 Model deset krat obdela besedilo.
Izguba podatkov: 4.1736
Natančnost: 0.4832
Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['say', 'for', 'the', 'be', 'was']

Test 5 Model dvajset krat obdela besedilo.
Izguba podatkov: 3.5199
Natančnost: 0.6339
Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['be', 'know', 'others', 'the', 'go']

Test 6 Model petdeset krat obdela besedilo.
Izguba podatkov: 2.9130
Natančnost: 0.7082
Poljuben stavek: It is hard enough to remember my opinions, without also remembering my reasons for them!
Sekvenca: It is hard enough to
Predvidene besede: ['it', 'the', 'for', 'â', 'but']

5.2 Rezultati

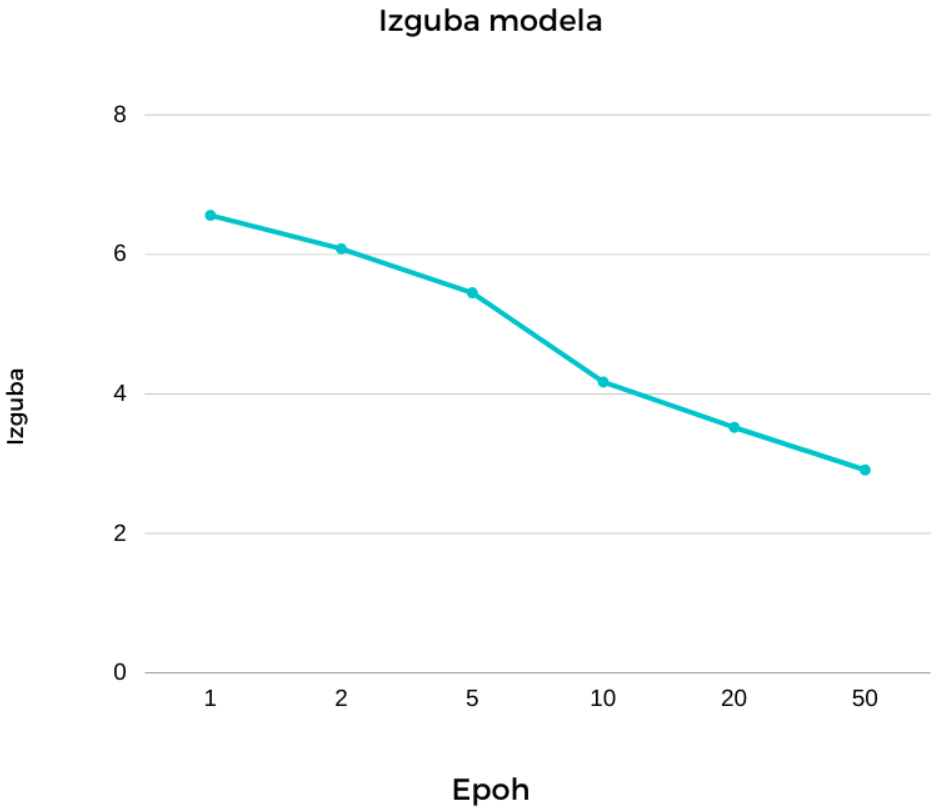
Natančnost Pri vsakem testiranju smo shranjevali natančnost in izgubo informacij tekom učenja. Prvi graf predstavlja zbrane podatke o natančnosti jezikovnega modela skozi čas. Os x predstavlja število epohov, ki smo jih izbrali za učenje, os y pa natančnost jezikovnega modela. Lepo je razvidno da črtovni graf od učenja z enim epohom lepo naraste do učenja z dvajsetimi in sicer na intervalu med 5 in 20 epohimi vidimo največji skok v natančnosti podatkov. Pokazatelj dobrega delovanja modela so tudi predikcije na tem intervalu, saj so skoraj vse rešitve kontekstualno pravilne. Nato pa graf začne počasneje naraščati, saj je količina podatkov večja in je potrebno vedno več učenja za višjo natančnost. To je tudi vidno na rezultatih predikcije postajajo vedno bolj zapletene in morda zaradi količine samih podatkov neveč tako pravilne.



Slika 4: Graf prikazuje razvijanje natančnosti skozi čas

Izguba podatkov Drugi graf pa predstavlja izgubo informacij med učenjem jezikovnega modela. Os x predstavlja količina izgubljenih podatkov, os y, pa enako kot pri prvem grafu, prikazuje izbrano število epohov. Iz grafa lahko razberemo, da je pri učenju z enim epohom izguba največja. Nato pa izguba po uporabi večih epohov

za obdelavo besedila vstrajno pada in nekako lahko ponovno opazimo, da je največje izboljšanje na intervalu med 5 in 20 epohi.



Slika 5: Graf prikazuje razvijanje izgube podatkov skozi čas

6 Zaključek

Delo predstavlja pregled konceptov jezikovnega modeliranja in implementacijo rekurzivnih nevronskih mrež za predvidevanje nadaljevanja besedila, ki imajo zaradi svoje sposobnosti učenja na zelo velikih količinah besedila in pomnenja preteklih stanj velik potencial na področju jezikovnega modeliranja.
Implementacija jezikovnega modela je bila uspešna, saj smo kot rezultat prejeli zelo smiselna nadaljevanja podane sekvence, vendar je tu še veliko prostora za izboljšave. Nadaljnje delo bi tako lahko vključilo tudi kako okolje vpliva na samega uporabnika, njegovo izbiro podanih predvidenih možnosti in kako prediktivni jezikovni modeli vplivajo na pisanje vsebine. Od izboljšanja natančnosti in manjše izgube podatkov pri učenju modela do napredka na področju strojne opreme, ki bi omogočila rekurzivnim nevronskim mrežam hitrejše učenje, manjka še funckija izbire domene, kjer bi za posamezne domene naučili nevronske mreže in nato bi pri uporabi jezikovnega modela omogočili samodejno ali pa ročno izbiranje domen, kar bi nam omogočilo uporabo željene domene pri katerem koli tipu pisanega besedila. Tako bi rešili problem manjkajočih besed v naučenem slovarju jezikovnega modela, saj bi v primeru pisanja znanstvenega besedila uporabili model naučen z znanstevnim besedilom ne pa model naučen z leposlovnim besedilom.
Še posebej pa bi prihodnje delo lahko osredotočili na področje kako bi podani predlogi lahko bili zasnovani tako, da bi slednji imeli zaželene učinke na podano vsebino. Na primer, s pomočjo predvidevanja nadaljevanja vsebine bi lahko piscem drugega jezika pomagali pisati bolj tekoče in naravno v izbranem jeziku, kar bi še bolj razširilo že tako široko uporabo jezikovnega modeliranja.

7 Literatura in viri

[1] Croft, W B in Ponte, J M, A Language Modeling Approach to Information Retrieval. *Research and Development in Information Retrieval, 1998*, sprejeto v objavo. *(Citirano na strani 2.)*

[2] Chen, Stanley in Goodman,Joshua, An Empirical Study of Smoothing Techniques for Language Modeling. *Harvard University, 1998*, sprejeto v objavo. *(Citirano na strani 2.)*

[3] Chang, Ming-Wei in Devlin, Jacob, Lee, Kenton. *Toutanova, Kristina* , sprejeto v objavo. Pre-training of Deep Bidirectional Transformers for Language Understanding,2018 *(Citirano na strani 3.)*

[4] Bengio, Yoshua, Neural net language models. *Scholarpedia, 2008*, sprejeto v objavo. *(Citirano na strani 3.)*

[5] De Mori, Renato in Kuhn, Roland, A cache-based natural language model for speech recognition. *EEE transactions on pattern analysis and machine intelligence,1990*, sprejeto v objavo. *(Citirano na strani 5.)*

[6] Madsen, Mathias Winther, The Limits of Machine Translation. *University of Copenhagen, 2009*, sprejeto v objavo. *(Citirano na strani 5.)*

[7] Charniak, Eugene, Statistical Techniques for Natural Language Parsing. *AI Magazine 18(4):33–44, 1997*, sprejeto v objavo. *(Citirano na strani 6.)*

[8] Masaru Tomita, Generalized LR Parsing. *Springer Science Business Media,2012*, sprejeto v objavo. *(Citirano na strani 7.)*

[9] Pham, Vu, Dropout improves recurrent neural networks for handwriting recognition. *14th International Conference on Frontiers in Handwriting Recognition,2014*, sprejeto v objavo. *(Citirano na strani 8.)*

[10] Jansen, B. J in Rieh, S., The Seventeen Theoretical Constructs of Information Searching and Information Retrieval. *Journal of the American Society for Information Sciences and Technology,2010*, sprejeto v objavo. *(Citirano na strani 8.)*

[11] Hopfield, J. J, Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences,1982*, sprejeto v objavo. *(Citirano na strani 9.)*

[12] Dupond, Samuel, A thorough review on the current advance of neural network structures. *Annual Reviews in Control,2019*, sprejeto v objavo. *(Citirano na strani 10.)*

[13] Williams, Ronald J in Zipser, David, A learning algorithm for continually running fully recurrent neural networks. *Neural computation,1989*, sprejeto v objavo. *(Citirano na strani 11.)*

[14] Goller, Christoph in Küchler, Andreas, Learning task-dependent distributed representations by backpropagation through structure. *International Conference on Neural Networks,1996*, sprejeto v objavo. *(Citirano na strani 11.)*

[15] Gers, Felix A. in Schmidhuber, Jürgen, Schraudolph, Nicol N.. *Learning task-dependent distributed representations by backpropagation through structure*, sprejeto v objavo. Journal of machine learning research,2002 *(Citirano na strani 11.)*

[16] Schmidhuber, J, Learning complex, extended sequences using the principle of history compression. *Neural Computation,1992*, sprejeto v objavo. *(Citirano na strani 12.)*

[17] He, Kaiming in Ren, Shaoqing, Sun, Jian. *Zhang, Xiangyu*, sprejeto v objavo. Learning task-dependent distributed representations by backpropagation through structure Proceedings of the IEEE conference on computer vision and pattern recognition,2016 *(Citirano na strani 13.)*

[18] Bengio,Yoshua in Mikolov,Tomas, Pascanu, Razvan. *On the difficulty of training recurrent neural networks*, sprejeto v objavo. International conference on machine learning,2013 *(Citirano na strani 13.)*

[19] Auli,Michael in Galley, Michel, Quirk, Chris. *Zweig, Geoffrey*, sprejeto v objavo. Joint language and translation modeling with recurrent neural networks In Proceedings of the ACL Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1044–1054,2013 *(Citirano na strani 14.)*

[20] Daeyoung Jang in Wootaek Lim, Taejin Lee. *Joint language and translation modeling with recurrent neural networks*, sprejeto v objavo. Asia-Pacific signal and information processing association annual summit and conference (APSIPA),2016 *(Citirano na strani 14.)*

[21] Lidong Bing in Piji Li, Wai Lam. *Zihao Wang*, sprejeto v objavo. Deep Recurrent Generative Decoder for Abstractive Text Summarization arXiv preprint arXiv:1708.00625,2017 *(Citirano na strani 14.)*

[22] Kuhlman, Dave, A python book: Beginning python, advanced python, and python exercises, 2009.

, sprejeto v objavo.22

[23] Venners, Bill, The Making of Python: A Conversation with Guido van Rossum, Part I. *Artima Developer,2012*, sprejeto v objavo. *(Citirano na strani 18.)*

[24] Chollet, François, Xception: Deep learning with depthwise separable convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition,2017*, sprejeto v objavo. *(Citirano na strani 18.)*

[25] Ng, Andrew, Machine Learning and AI via Brain simulations. *Accessed: May,2013*, sprejeto v objavo. *(Citirano na strani 19.)*

[26] Bressert, Eli, SciPy and NumPy: an overview for developers. *O'Reilly Media, Inc.,2012*, sprejeto v objavo. *(Citirano na strani 19.)*

[27] Bressert, Eli in Harris, David, Digital design and computer architecture. *Morgan Kaufmann,2010*, sprejeto v objavo. *(Citirano na strani 19.)*

[28] dos Santos, Leonardo Araujo, Artificial Inteligence. *GitBook,2017*, sprejeto v objavo. *(Citirano na straneh VII in 10.)*

[29] Carrier, Pierre Luc in Cho, Kyunghyun, LSTM networks for sentiment analysis. *Deep Learning Tutorials,2014*, sprejeto v objavo. *(Citirano na straneh VII in 12.)*