Mestna občina Celje

Komisija Mladi za Celje

ALI JE LAHKO INTERNET PRIJAZEN?

Nevronske mreže za pomoč pri pisanju prijaznih spletnih komentarjev

Raziskovalna naloga

AVTOR:

Drejc Pesjak

MENTOR:

Dušan Fugina, univ. dipl. inž.

Žalec, marec 2018

ALI JE LAHKO INTERNET PRIJAZEN?

Nevronske mreže za pomoč pri pisanju prijaznih spletnih komentarjev

Raziskovalna naloga

Avtor:
Drejc Pesjak, R4a

Mentor:
Dušan Fugina, univ. dipl. inž.

Mestna občina Celje, Mladi za Celje

Žalec, marec 2018

1 KAZALO

KAZALO SLIK

Povzetek

Dandanes je na spletu vedno več sovražnega govora, kamor vključujemo tudi rasizem, šovinizem, homofobijo … Poleg sovražnega najdemo tu še žaljiv govor, kot je klevetanje in žaljenje. Na večini spletnih portalov se ta problem rešuje z moderiranjem uporabnikovih komentarjev, največkrat z izbrisom. Tukaj naletimo na drug problem. Zatiranje svobode govora.

Avtomatizirano odkrivanje žaljivega govora pa to težavo le še poslabša.

V tej raziskovalni nalogi sem poskušal omenjene probleme rešiti z rekurentno nevronsko mrežo, ki neprimerne komentarje ne le označi, temveč jih tudi popravi, popravek pa je nato kot predlog vrnjen avtorju komentarja. Mreža v času pisanja raziskovalne naloge slabo posplošuje na še ne videne primere, zaradi premajhne količine učnih podatkov.

Ključne besede

Sovražni govor, žaljiv govor, moderatorstvo, strojno učenje, nevronske mreže, sequence to sequence nevronska mreža, podatkovno rudarjenje, računalništvo v oblaku.

Abstract

Nowadays there is an increasing number of hate speech on the Internet, including racism, chauvinism, homophobia ... In addition to the hate, there is also the pressence of offensive language, such as cursing and insults. Most of the online news websites solve this problem by modifying user comments, most often by deletion. Here we encounter another problem. Violation of freedom of speech.

Automated detection of offensive speech only makes this problem worse.

In this research I tried to solve the problems mentioned above with a sequence to sequence neural network that not only marks inappropriate comments, but also corrects them. Afterwards the correction is returned as a suggestion to the author of the comment. At the time of writing, the network gently exaggerates the unseen cases, due to the insufficient amount of learning data.

Keywords

Hate speech, offensive language, moderation, machine learning, neural networks, sequence to sequence neural network, data mining, cloud computing.

Kratice

- UI – umetna inteligenca
- UNM – umetna nevronska mreža.
- NLP – natural language processing - procesiranje naravnega jezika
- IBM - International Business Machines (podjetje)
- Csv - comma-separated values - vrednosti ločene z vejico
- Txt – končnica besedilne datoteke.
- IDLE - integrated development environment za Python – integrirano razvojno okolje za Python.
- API - aplikacijski programski vmesnik
- HTML – HyperText Markup Language
- CSS – Cascade Style Sheet
- SQL – Structured Query Language

2 UVOD

2.1 PREDSTAVITEV RAZISKOVALNEGA PROBLEMA

Starost otrok, ki segajo po pametnih napravah, se iz generacije v generacijo manjša. S tem dobivajo dostop do raznovrstnih spletnih vsebin, tako primernih kot neprimernih. Slabih informacij, kot je sovražni govor, se, predvsem zaradi njihove mentalne nerazvitosti, ne morajo obraniti in tako, kot majne spužve srkajo vase vse vsebine, ki jim pridejo pod prste. Takšna izpostavljenost negativnim vplivom spreminja generacije v ljudstvo pesimističnih in sovražnih ljudi, katerih besednjak sestavljajo primarno slabe besede.[16]

Način preprečevanja teh slabih vplivov vidim skozi moderatorstvo spletnih komentarjev na novičarskih straneh, kot so siol.net, 24ur.com, rtvslo.si, dnevnik.si … Problem, s katerim se sooča večina le-teh, je poplava komentarjev, ki jih moderatorji s težavo pregledujejo. Na spletnih časopisih, kot je New York Times, dobijo dnevno tudi po več deset tisoč komentarjev na članek, zaradi česar imajo omogočeno komentiranje le prvih 24 ur, pri člankih, o spornih temah, pa je komentiranje popolnoma onemogočeno.

Brisanje sovražnih komentarjev nas pripelje do nove težave, ki je v nasprotju z ideologijo spleta. Splet naj bi bil svobodno mesto, ki omogoča vsakemu posamezniku, ne glede na starost, spol ali raso, da izrazi svoje mnenje, četudi je v nasprotju z mišljenjem množice. Boljša možnost kot brisanje bi bila priporočena istopomenska različica komentarja v prijaznejši izvedbi. Moderator mora poleg komentarja poznati tudi vsebino članka, saj se za neprimernega upošteva tudi komentar, ki se ne navezuje na vsebino.

2.2 NAMEN RAZISKOVALNE NALOGE

Glede na to, da je spletnih komentarjev tako veliko, njihovo število pa se zaradi porasta spletnih uporabnikov veča, bi bilo smiselno avtomatizirati proces izločanja žaljivih komentarjev oziroma popravljanja le-teh. Zato sem se odločil uporabiti najnovejšo tehnologijo, ki je optimizirana za delovanje s tolikšno količino podatkov – nevronske mreže. Namen moje nevronska mreže je, ne le razločevati med primernimi in neprimernimi komentarji, ampak komentarje tudi izboljšati.

Cilj je naučena nevronska mreža, ki jo bo lahko administrator spletnega mesta preko programske knjižnice vključil v svojo spletno stran. Vsak komentar, ki ga bo uporabnik vnesel, bo posredovan nevronski mreži in po potrebi vrnjen uporabniku, ki bo komentar potrdil ali zavrnil.

2.3 HIPOTEZE

Za usmerjeno delo sem si zastavil nekaj hipotez, ki jih bom poskušal potrditi oziroma zavreči.

1. Delo moderatorja je možno avtomatizirati.
2. Žaljive komentarje je možno izboljšati.
3. Vsebina članka izboljša natančnost modela strojnega učenja.
4. Z nevronsko mrežo lahko komentar naredimo prijazen.

Prvo hipotezo bo mogoče potrditi, v kolikor bo natančnost modela presegala izbrano mejo 85 odstotkov, o drugi hipotezi pa bo odločalo število popravljenih komentarjev v odvisnosti s tistimi, pri katerih izboljšave niso bile mogoče.

Za pravilno ocenitev tretje hipoteze bo potrebno učenje nevronske mreže, na dveh učnih množicah, pri čemer ena ne vsebuje besedil člankov. S primerjanjem natančnosti bomo lahko ugotovili, ali je besedilo članka res ključna komponenta pri moderiranju komentarjev.

Četrta hipoteza je odvisna od druge, kar pomeni, da jo je možno potrditi le pod pogojem, da druga hipoteza drži. Pomembno je še omeniti, da obratna odvisnost ne velja. Prav tako se navezuje na prvo hipotezo, kjer gre za enako korelacijo kot med četrto in drugo.

3 TEORETIČNA ZASNOVA

3.1 ŽALJIVI IN SOVRAŽNI GOVOR

V vsakodnevni rabi enačimo sovražni, žaljiv in neprimeren govor, čeprav gre za tri različne kategorije komunikacije.

Sovražni govor je kazniv. Pomeni izražanje v večini primerov diskriminatornega mnenja, kot je rasna, spolna ali etnična diskriminacija. Cilj sovražnega govora je poniževanje in razvrednotenje pripadnikov določene manjšine. Izražen je lahko govorno, pisno ali nebesedno in gre za kaznivo dejanje s pravno podlago za pregon. V Sloveniji je možno anonimno prijaviti takšno vrsto sovražnega govora na portalu spletno-oko.si, ki ob domnevi kršitve 297. člena kazenskega zakonika prijavo posreduje policiji.

Žaljiv govor ima širši pomen, ki med drugim zajema »primere, kot so obrekovanje, grožnje, razžalitve, krive obdolžitve, škodljivost za otroke ipd.«[27]

Tretja vrsta je družbeno neprimerna komunikacija, ki je po družbenih neformalnih normah nesprejemljiva za neko skupino ljudi. Toleranca se pri tej kategoriji od posameznika do posameznika dovolj opazno spreminja. »Gre predvsem za kletvice, opolzke besede, nestrpnost, zmerjanje, določene žalitve verskih čustev, pa tudi zgolj za primitivno, nedostojno ali nespodobno rabo slovenskega jezika.«[27]

3.2 STROJNO UČENJE

Umetna inteligenca je posnemanje procesov človeške inteligence v strojih. Med procese spadajo:

- razumevanje oziroma uporaba navodil za doseganje ciljev,
- samo-izboljševanje in
- učenje, pri čemer gre za zbiranje podatkov za kasnejšo uporabo.

Poznamo dve vrsti umetne inteligence. Medtem, ko je ozka umetna inteligenca (narrow artificial intelligence) specializirana za naloge na enem področju, je splošna umetna inteligenca sposobna izvajanja vseh človeških procesov. Sposobnost učenja je dosežena z različnimi algoritmi brez izrecnega programiranja posameznih pravil. Model se na podlagi preteklih izkušenj, z učenjem na vzorčnih podatkih, nauči vzorce in je nato zmožen s posplošitvijo napovedovati.

Začetki strojnega učenja segajo v leto 1950, ko je Alan Turing zasnoval Turingov test za prepoznavanje dovršenosti modela. Takratni problem je bilo počasno procesiranje podatkov predvsem pa primanjkljaj le-teh, kar je omogočilo razmah področja v zadnjih parih letih. Poleg Turingovega testa lahko v zgodovini strojnega učenja izpostavimo še druge dosežke, kot so: IBM-ov program Deep Blue, ki je leta 1990 premagal šahovskega velemojstra Gary Kasparova, ter super računalnik Watson, ki je zmagal v televizijski oddaji kviza Jeopardy, impresivna pa je tudi zmaga Googlovega projekta AlphaGo v najtežji igri na svetu Go.

Danes veliko večjih podjetij uporablja strojno učenje za izboljšanje poslovanja, uporablja pa se tudi za procesiranje naravnega jezika, prepoznavanje govora, strojni vid ...

3.3 NADZOROVANO UČENJE

Strojno učenje lahko razdelimo v tri podkategorije: nadzorovano, nenadzorovano in okrepljeno učenje.

Pri nadzorovanem učenju gre za organizirane podatke z označenimi primerki. Model se tako skozi fazo učenja prilagodi podatkom, do te mere, da je z visoko natančnostjo zmožen za dan vhodni primerek pravilno določiti oznako le tega.

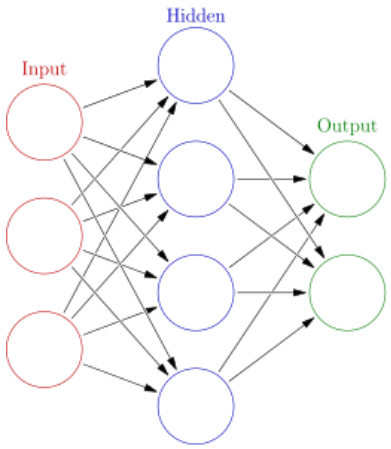
Nenadzorovano učenje se razlikuje po strukturi podatkov, ki so neoznačeni, s tem pa tudi končen rezultat ni viden. Kar pomeni, da mora model najti ponavljajoče se vzorce v podatkih in jih razvrstiti v avtomatske razrede.

Okrepljeno učenje se uporablja v simuliranih ali resničnih okoljih, kjer agent poskuša doseči določen cilj in izboljšati izvedbo s pomočjo pozitivnih ali negativnih spodbud. Agent ne pozna naslednjega najboljšega koraka, kar pomeni, da lahko doseže izboljšanje le s poskušanjem.

3.4 UMETNE NEVRONSKE MREŽE

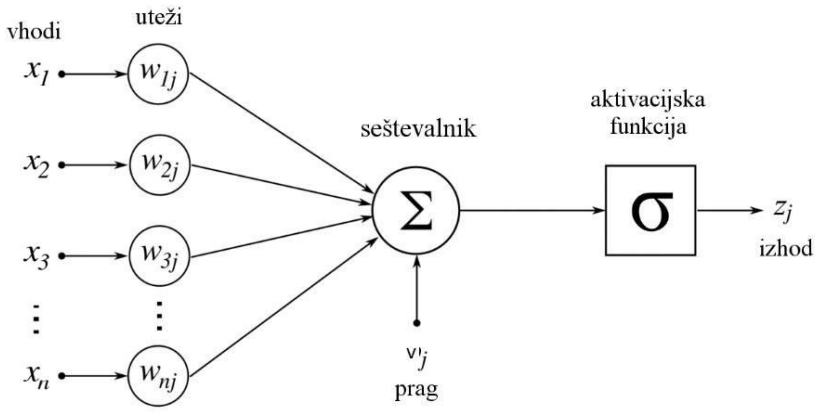
Kot je bilo že prej omenjeno, poskušamo s temi algoritmi doseči človeku podobno obnašanje. Od tod tudi inspiracija za glavni element umetnih nevronskih mrež – nevron. Tako kot človeške so tudi umetne nevronske mreže sestavljene iz nevronov, ki pošiljajo signale in se prožijo ob prejemanju.

Struktura nevronske mreže je običajno sestavljena iz več plasti medsebojno povezanih nevronov. Prva se imenuje vhodna, zadnja pa izhodna, obe sta tudi omejeni na velikost vhodnih oziroma izhodnih podatkov. Vse vmesne plasti so skrite in skrbijo za izvajanje računskih operacij, kjer vsaka plast razbere drugačno lastnost vhodnih podatkov.



SLIKA 1: STRUKTURA NEVRONSKE MREŽE

Proces nevronovega delovanja se zgodi v štirih glavnih korakih. Najprej je prejemanje podatkov, kjer procesna enota pridobi vhodne informacije (x_1, x_2, x_3, …, x_n). Sledi množenje vsakega vhodnega podatka z ustreznimi utežmi (w_1, w_2, w_3, …, w_n). Na seštevku se nato izvede aktivacijska funkcija (sigmoidna ali koračna), ki vrne izhodni rezultat (z_j).

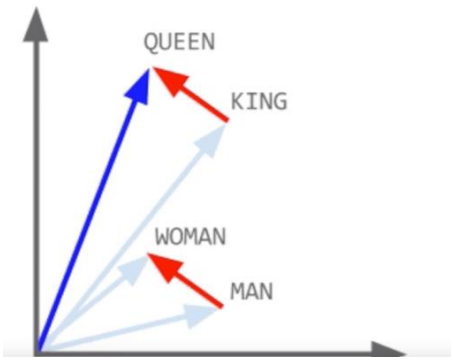


SLIKA 2: SESTAVA UMETNEGA NEVRONA

Pri učnem procesu so uteži skoraj naključno inicializirane. Po vsakem primerku, poslanem skozi plasti nevronske mreže, dobimo rezultat, ki ga nato primerjamo s pravilno oznako, ki je priložena v učni množici. Med rezultatom in odgovorom izračunamo napako (razlika med pričakovano in dejansko vrednostjo). Da bi napako zmanjšali, prilagodimo uteži in postopek ponovimo.

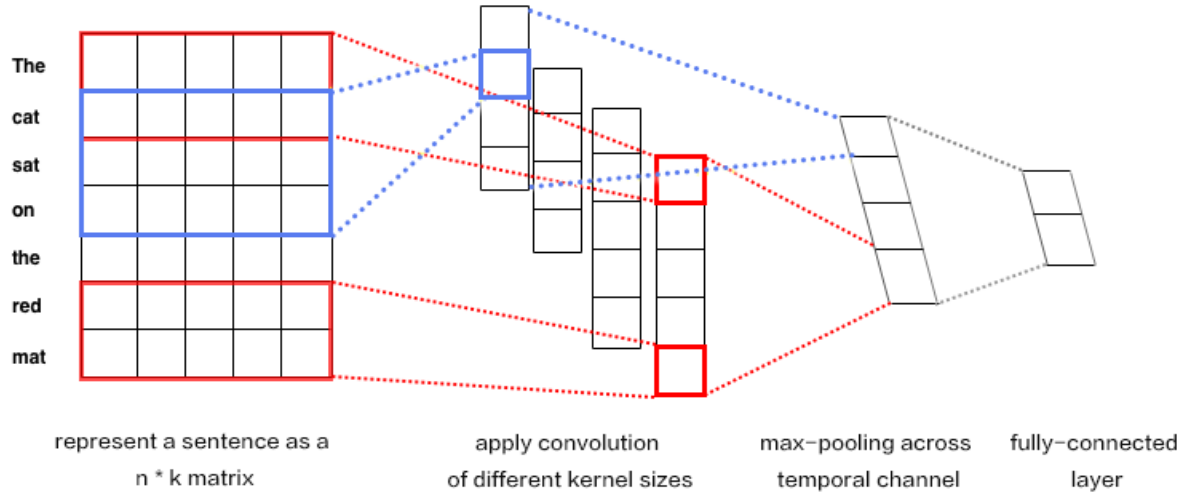
3.5 KONVOLUCIJSKE NEVRONSKE MREŽE

Posebna vrsta nevronske mreže je konvolucijska. Po navadi ima od 5 do 25 slojev, njeno delovanje pa imitira človeški vid. V večini primerov je vhodni podatek v številke spremenjena slika, pri naravnem procesiranju jezika pa so to vektorizirane besede, kjer vsaka beseda predstavlja eno vrstico mreže enako dolgih vektorjev.



SLIKA 3: VEKTORIZACIJA BESED

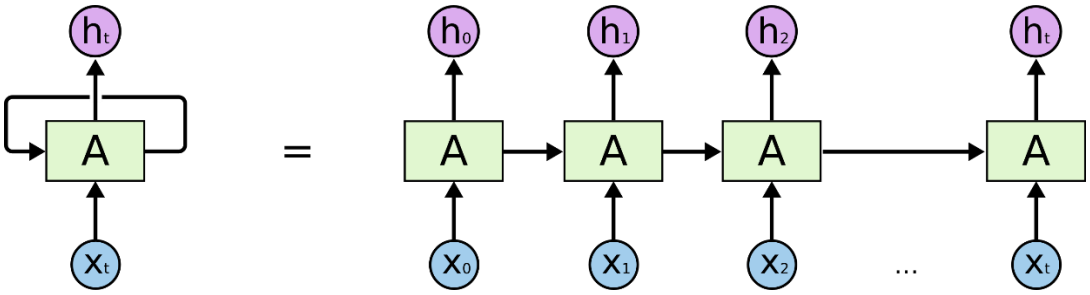
Konvolucijsko nevronsko mrežo sestavljajo trije različni sloji: konvolucijski sloj s funkcijo ReLu, združevalni sloj in polno povezan sloj, pri čemer globlji sloji razberejo bolj abstraktne lastnosti. Prvi (konvolucijski) sloj s premikanjem različnih filtrov preko vhodne mreže podatkov razbere različne lastnosti, pri sliki so to robovi in oblike. Združevalna plast sliko poenostavi, kar pomeni, da spregleda hrup (nepotrebno) in prepreči pretirano prilagajanje. S tem izgubimo informacijo o prostorski relaciji med različnimi lastnostmi. Na koncu takšne nevronske mreže so polno povezani sloji, kjer je, tako kot v običajni UNM, vsak nevron v sloju povezan z vsakim iz naslednjega. Polno povezan sloj združi lastnosti, naučene z različnimi filtri, in služi kot klasifikator, medtem ko konvolucijski in združevalni sloj delujeta kot ekstraktorja lastnosti.



SLIKA 4: KONVOLUCIJSKA NEVRONSKA MREŽA ZA NLP

3.6 REKURENTNE NEVRONSKE MREŽE

Rekurentne nevronske mreže so z razliko zank enake navadnim. Omogočajo obdelavo sekvenčnih podatkov, kot je na primer besedilo, pri čemer je en primerek ena beseda ali celo samo en znak. Poleg besede (x_t) je vhodni podatek še stanje iz prejšnjih časovnih korakov.



SLIKA 5: RAZVITA REKURENTNA NEVRONSKA MREŽA

Problem predstavlja kratkoročen spomin, ki ga te mreže imajo, saj z vsakim korakom mreža izgublja informacije o podatkih iz zgodnjih korakov. Ta problem rešuje posebna vrsta imenovana Long-Short Term Memory (LSTM). LSTM ima t. i. vrata (pozabljalna, vhodna in izhodna), s katerimi določi kateri podatki iz prejšnjih celic so pomembni, da jih nato upošteva pri računanju izhoda. Podoben LSTM je GRU, ki vsebuje le dvoje vrat (posodobitvena in ponastavitvena), kar pomeni, da je njegova zgradba manj kompleksna in je zato računsko bolj učinkovit.

3.6.1 REKURENTNE NEVRONSKE MREŽE ZA MODELIRANJE SEKVENC

Rekurentne nevronske mreže v vsakem koraku poskušajo napovedati naslednji znak/besedo, kar jim omogoča, da po končanem učenju ustvarijo besedilo v slogu učnih podatkov. Nekoliko drugačen namen imajo mreže za modeliranje sekvenc, ki so sestavljene iz dveh LSTM mrež. Iz vhodnega besedila, prejetega besedo za besedo, kodirnik proizvede miselni vektor. Ta vektor, ki zajema bistvo besedila in se ne oklepa na način zapisa, je nato posredovan dekoderju. Dekoder spremeni vektor v slovnično pravilen odgovor, tako kot prej besedo za besedo oziroma znak za znakom.

4 EMPIRIČNI DEL

4.1 POTREBNO ZNANJE IN IZBIRA ORODIJ

Ob zastavitvi problema sem kaj kmalu ugotovil, da moje trenutno znanje ne bo zadostovalo. Kljub vsemu znanju v različnih računalniških jezikih, kot so C#, Java, HTML, SQL, CSS ... pridobljenem skozi zadnja štiri leta šolanja in osnovam v jeziku Python, sem bil slabo podkovan na področju nevronskih mrež. Področje nevronskih mrež zahteva tudi zelo dobro poznavanje matematike.

Svoje znanje sem dopolnil na spletni strani datacamp.com, kjer sem dokončal tečaje, kot so Uvod v Python za podatkovno znanost, Nadaljevanje v Pythonu za podatkovno znanost in Strojno učenje v Pythonu. Znanje sem pridobival tudi z branjem mnogih pojasnjevalnih člankov in raziskovalnih nalog s tega področja. V največjo pomoč pa so mi bili posnetki na portalu YouTube ter seznami predvajanja Učenje Pythona za podatkovno znanost, Strojno učenje za hekerje ter Uvod v globoko učenje, katerih avtor je Siraj Raval.

Programski jezik Pyhton 3.6 sem uporabil za programiranje nevronskih mrež ter obdelavo podatkov in delo z različnimi formati datotek (.csv in .txt). Za pisanje kode sem uporabljal program Python IDLE in Python Shell za zaganjanje skript. Csv dokumente sem pregledoval v programih Excel in NotePad++ ter z Okensko ukazno vrstico naložil manjkajoče Python knjižnice.

4.2 UPORABLJENE TEHNOLOGIJE

Globoko učenje je eno izmed najnovejših področij, zaradi česar sem se odločil uporabiti novejšo verzijo 3.6 programskega jezika Python. Drugi razlog so bile knjižnice, kot so TensorFlow in Keras, ki preprosto niso kompatibilne s starejšo različico. Poleg knjižnic TensorFlow in Keras sem uporabljal še NumPy, re, Pandas in os.

4.2.1 RAZLIČNI VMESNIKI OPERACIJSKEGA SISTEMA

Os je knjižnica za programski jezik Python, z uvoznim imenom os. Ta modul omogoča uporabljanje funkcionalnosti operacijskega sistema. Z ukazom open lahko v datoteko pišemo ali pa iz nje beremo podatke, z nadaljnjim ukazom readlines celo vse vrstice datoteke hkrati. Omogoča nam tudi delo s potmi in ustvarjanje novih in začasnih imenikov.

4.2.2 OPERACIJE REDNEGA IZRAŽANJA

Knjižnica z uvoznim imenom re, je namenjena za delo z nizi. Z različnimi ukazi je možno v besedilu poiskati vzorčni niz in ga tudi zamenjati z drugim. Pred posebnimi znaki je potrebno postaviti poševnico ali pa niz označiti za surovega s predpono r.

4.2.3 NUMPY

NumPy je Pythonova knjižnica za obdelavo velikih večdimenzionalnih seznamov in matrik ter ponuja veliko kolekcijo matematičnih funkcij s področja linearne algebre, ki omogočajo učinkovito računanje s prej omenjenimi strukturami. Glavna funkcionalnost je

n-dimenzionalen seznam (ndarray), ki, za razliko od v Python vgrajenega dinamičnega seznama, lahko vsebuje le elemente istega podatkovnega tipa.

4.2.4 PANDAS

Za podatkovno analitiko in delo s podatki je pogosto uporabljena odprtokodna knjižnica Pandas, ki ponuja podatkovne strukture in orodja za manipuliranje numeričnih tabel. Uporablja posebno strukturo imenovano DataFrame in med drugim omogoča branje .csv datotek ter pametno rezanje seznama podatkov bodisi po oznakah ali indeksih. Pogosto uporabljena funkcionalnost je preoblikovanje $n \cdot m$ seznama v obliko $m \cdot n$.

4.2.5 TENSORFLOW

TensorFlow je Python knjižnica za hitro numerično računanje. Ustvarilo jo je podjetje Google in je namenjena tako za raziskave kot za proizvodnjo. Posebnost je, da jo je možno poganjati na centralni procesni enoti, grafični procesni enoti ali pa na telefonu. Ob instalaciji knjižnica TensorFlow že vključuje razne modele globokega učenja, ki jih lahko direktno uporabimo.

4.2.6 KERAS

Keras je visokonivojski aplikacijski programski vmesnik (API) za nevronske mreže, katerega osnova je prej omenjena knjižnica TensorFlow. Ta knjižnica za globoko učenje je napisana v Pythonu in omogoča enostavno in hitro izdelavo prototipov, tako konvolucijskih kot rekurentnih nevronskih mrež. Poleg tega pa brezhibno teče na CPE in GPE.

4.3 RAZISKAVE DRUGIH

V zadnjih parih letih, odkar so računalniki postali zelo zmogljivi in zato tudi to področje precej bolj zanimivo ter obetajoče, je že marsikdo poskusil svojo "srečo" v avtomatskem zaznavanju sovražnega in žaljivega govora na spletu. Raziskovalci so gradili na raziskavah drug drugega in skozi leta zviševali natančnost sistemov.

Kot je že poudaril Dirk Hovy[11], igrajo veliko vlogo tudi demografske značilnosti, kot so starost, spol, regija. Njegova odkritja potrjuje raziskava[36], ki jo je naredil z Zeerak Waseem na korpusu komentarjev s socialnega omrežja Twitter. Ugotovila sta, da na žaljivost komentarja vpliva spol (moški so odgovorni za večino sovraštva na spletu), medtem ko druge demografske komponente ne prikažejo nobenih izboljšanj. Nadaljnje raziskave[33] kažejo, da je ključnega pomena tudi ločevanje med sovražnim in žaljivim govorom. Takšno ločevanje ustvari jasnejše meje med razredi, kar omogoči doseči le pet odstotno napako pri določevanju sovražnega govora.

V svojem modelu sem moral upoštevati tudi nepovezane komentarje, ki lahko predstavljajo tudi do petindvajset odstotkov[35] odstranjenih komentarjev.

V povezavi z algoritmi za detekcijo sovražnega govora je v drugi raziskavi[21] bilo uporabljeno sedemnajst različnih modelov strojnega učenja, med katerimi sta se najbolje izkazala konvolucijska nevronska mreža in mreža LSTM. Kar je vplivalo tudi na mojo izbiro, konvolucijska mreža s 86 odstotno natančnostjo. Med slabše se je uvrstil model

imenovan Bag of Words, ki ne upošteva zaporedja, v katerem se besede pojavijo, ampak le, kolikokrat se te ponovijo v stavku.

4.4 ZBIRANJE PODATKOV

V podatkovni analitiki je zbiranje podatkov po navadi najzahtevnejši del oziroma vsaj časovno najbolj potraten. Čeprav je informacijska doba seboj prinesla ogromne količine podatkov, ki se iz leta v leto eksponentno večajo, so ti podatki v naravni obliki skoraj vedno nestrukturirani, lahko rečemo, da niso kvalitetni.

4.4.1 KOMENTARJI SPLETNIH PORTALOV

Najobetavnejši izmed novičarskih portalov je bil The Guardian, kjer je bilo videti, da neprimerne komentarje administratorji le blokirajo in se na mestu komentarja izpiše besedilo »Vaš komentar je moderator odstranil, ker ni ustrezal kriterijem skupnosti. Odgovori na komentar so prav tako lahko izbrisani. Za podrobnosti si oglej naša pravila (vprašanja in odgovori).« Ideja je bila, da bi preko njihovega APIja lahko prišel do teh blokiranih komentarjev. Za aplikacijski programski vmesnik je potrebno pridobiti ključ, bodisi razvojni ali komercialni, ki omogoča množično pošiljanje zahtev. Z ustreznim programom je možno upravljati API preko Python kode ter preko drugega pogovornega API dostopati do komentarjev člankov. Kasneje sem ugotovil, da komentarji niso blokirani, ampak odstranjeni ter da je do njih nemogoče priti.

Na ostalih spletnih straneh sem običajno naletel na eno izmed dveh situacij. Prva je onemogočeno komentiranje, bodisi zaradi prevelikega števila morebitnih komentarjev ali premajhnega števila moderatorjev, kar je prisotno na portalih, kot so The New York Times, ABC News, CNBC, CNN in NBC News. V drugem primeru gre za brisanje komentarjev na portalih The Washington Post, Fox News, BBC, The Verge, Huffington Post… Mednje spadajo tudi večje slovenske medijske hiše 24ur, siol.net in RTV SLO.

4.4.2 FACEBOOK IN TWITTER KOMENTARJI

Zaradi težav pri zbiranju podatkov z novičarskih strani sem se obrnil k alternativnim virom. Po dolgem iskanju sem naletel na zbirko objav s komentarji[13] zbranih, s socialnega omrežja Facebook. Objave so kot majhni izseki članka, ki ga skupaj s povezavo novičarski portali, kot je The Guardian, objavijo na Facebooku. Zbirka podatkov vsebuje 19850 objav iz 83 različnih novinarskih organizacij in osebnosti, ki predstavljajo zadnjih 250 strani objav, objavljenih do 14. julija 2017. Vsaka objava ima do 100 komentarjev za skupno 1.025.403 komentarjev.

Poleg prve zbirke, ki mi je služila kot glavni vir podatkov, sem uporabil še 25 tisoč komentarjev[7] iz Twitter-ja za učenje konvolucijske mreže prepoznavanja žaljivih komentarjev. Komentarji so bili razvrščeni v tri skupine (sovražen, žaljiv govor ali nič od tega), ki sem jih moral združiti v dve (žaljiv in nežaljiv govor).

4.5 RAZVRŠČANJE KOMENTARJEV

Odločil sem se uporabiti tekstovno konvolucijsko mrežo, z namenom da bi si olajšal delo pri razvrščanju in popravljanju komentarjev s Facebook-a. Osnovo sem našel na spletu, ki je bila priložena poučnemu članku. Ker je bila mreža učena na pozitivnih in negativnih kritikah filmov spletne podatkovne baze o filmih in televizijskih programih ter filmskih igralcih, imdb.com, so bili podatki ločeni v dve datoteki *rt-polarity.pos* in *rt-polarity.neg*. Nato sem natipkal program v jeziku Python, ki je komentarje iz Twitter zbirke ločil glede na žaljive in prijazne v prav tako dve različni datoteki z istim imenom, a v drugem imeniku.

splitTweetNegPos.py - G:\ShareLAPTOP\seminarska\python\splitTweetNegPos.py (3.6.2)

File Edit Format Run Options Window Help

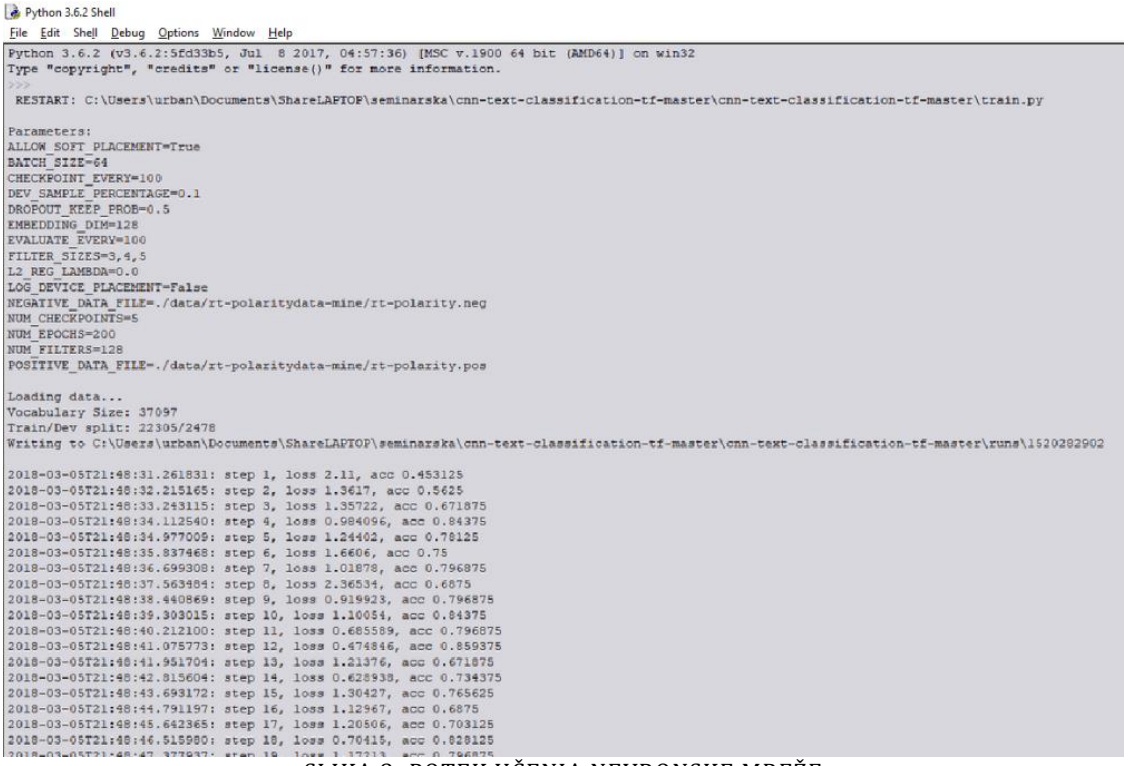
```
import pandas as pd

df = pd.read_csv("C:/Users/urban/Documents/ShareLAPTOP"
                 "/seminarska/hate-speech-and-offensive-language-master"
                 "/hate-speech-and-offensive-language-master/data"
                 "/organizedData.csv", encoding='utf8')


positive_tweets = []
with open("C:\\Users\\urban\\Documents\\ShareLAPTOP\\seminarska"
          "\\cnn-text-classification-tf-master\\cnn-text-classification-tf-master"
          "\\data\\rt-polaritydata-mine\\rt-polarity.neg", "w") as text_file:
    for i,t in df.iterrows():
        if t[0] == 0 or t[0] == 1:
            text_file.write(t[1]+'\n')
        elif t[0] == 2:
            positive_tweets.append(t[1])

with open("C:\\Users\\urban\\Documents\\ShareLAPTOP\\seminarska"
          "\\cnn-text-classification-tf-maste\\cnn-text-classification-tf-master"
          "\\data\\rt-polaritydata-mine\\rt-polarity.pos", "w") as text_file:
    for tweet in positive_tweets:
        text_file.write(tweet+'\n')
```

SLIKA 6: KODA ZA RAZPOREDITEV TWEETOV

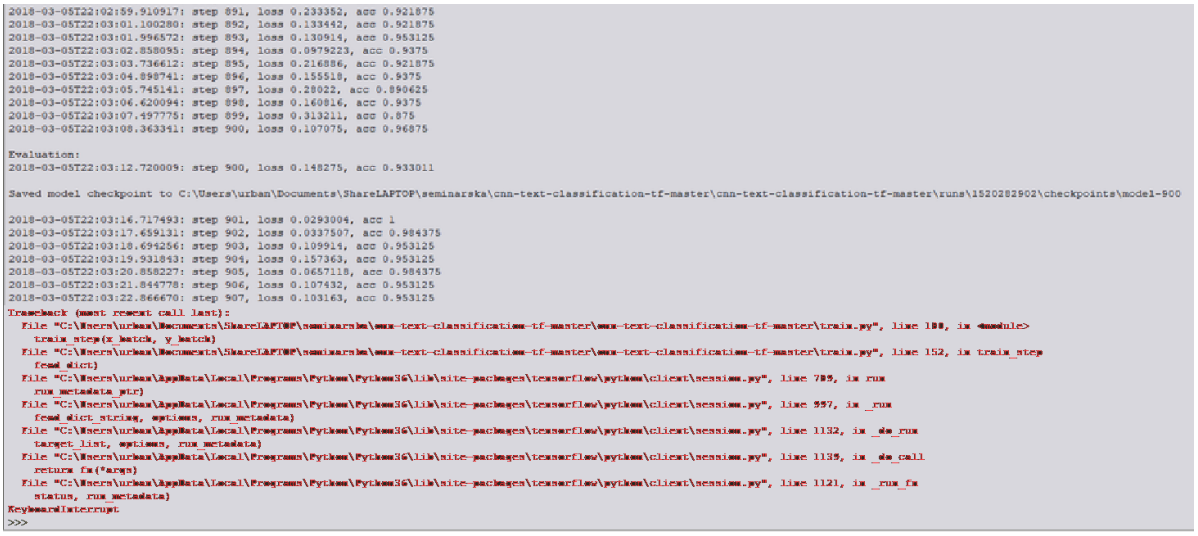
V skripti za učenje mreže sem spremenil pot do datotek in zagnal program *train.py*.

```
# Data loading params
tf.flags.DEFINE_float("dev_sample_percentage", .1, "Percentage of the training data to use for validation")
tf.flags.DEFINE_string("positive_data_file", "./data/rt-polaritydata-mine/rt-polarity.pos", "Data source for the positive data.")
tf.flags.DEFINE_string("negative_data_file", "./data/rt-polaritydata-mine/rt-polarity.neg", "Data source for the negative data.")
```

SLIKA 7: SPREMENJENA POT DO UČNIH PODATKOV



SLIKA 8: POTEK UČENJA NEVRONSKE MREŽE



SLIKA 9: SHRANJEVANJE PARAMETROV IN PREKINITEV PROGRAMA

Program je po vsakih sto korakih shranil »checkpoint« oziroma naučene parametre mreže. Po devetstotih korakih pa sem program prekinil, da se nevronska mreža ne bi preveč prilagajala.

Sledila je obdelava Facebook komentarjev, kjer sem moral izluščiti le stolpec s komentarji in jih očistiti znakov \r in \n, ki označujeta novo vrstico. Odstranil sem tudi vse smeškote in druge ne ASCII znake. Podatke sem nato shranil pod imenom *rt-polarityFB.pos* v drugačnem imeniku, kot prej.

SLIKA 10: OBDELAVA FACEBOOK KOMENTARJEV

S komentiranjem dela kode, v datoteki *data_helpers.py*, sem onemogočil obdelavo druge datoteke (datoteka z negativnimi komentarji) in v datoteki *eval.py* z enakim postopkom odstranil delovanje funkcije za izračun natančnosti. V drugi datoteki sem tako kot prej še spremenil pot do datoteke s komentarji.

```
def load_data_and_labels(positive_data_file, negative_data_file):
    """
    Loads MR polarity data from files, splits the data into words and generates
    Returns split sentences and labels.
    """
    # Load data from files
    positive_examples = list(open(positive_data_file, "r").readlines())
    positive_examples = [s.strip() for s in positive_examples]
##    negative_examples = list(open(negative_data_file, "r").readlines())
##    negative_examples = [s.strip() for s in negative_examples]
    # Split by words
    x_text = positive_examples # + negative_examples
    x_text = [clean_str(sent) for sent in x_text]
    # Generate labels
    positive_labels = [[0, 1] for _ in positive_examples]
    negative_labels = [[1, 0] for _ in positive_examples]
    y = np.concatenate([positive_labels, negative_labels], 0)
    return [x_text, y]
```

SLIKA 11: ODSTRANITEV PRAZNIH SPREMENLJIVK

```
# Print accuracy if y_test is defined
##if y_test is not None:
##    correct_predictions = float(sum(all_predictions == y_test))
##    print("Total number of test examples: {}".format(len(y_test)))
##    print("Accuracy: {:g}".format(correct_predictions/float(len(y_test))))
```

SLIKA 12: ODSTRANITEV FUNKCIJE ZA IZRAČUN NATANČNOSTI

Nato sem odprl Windowsovo ukazno vrstico ter se z ukazom *cd C:\pot* premaknil v mapo, kjer se nahaja program *python.py* za poganjanje skript. Nato sem, z ukazom *python C:\pot_do_programa\eval.py –eval_train* pognal program nevronske mreže za napovedovanje oznak komentarjev, ki so se shranile, skupaj s komentarji, v datoteko *prediction.csv*.

```
C:\Users\User\AppData\Local\Programs\Python\Python36>python C:\Users\User\Desktop\seminarska\cnn-text-classification-tf-master\cnn-text-classification-tf-master\eval.py --eval_train

Parameters:
ALLOW_SOFT_PLACEMENT=True
BATCH_SIZE=64
CHECKPOINT_DIR=C:/Users/User/Desktop/seminarska/cnn-text-classification-tf-master/cnn-text-classification-tf-master/runs/1520282902/checkpoints/
EVAL_TRAIN=True
LOG_DEVICE_PLACEMENT=False
NEGATIVE_DATA_FILE=./data/rt-polaritydata-mine/rt-polarityFB.neg
POSITIVE_DATA_FILE=C:/Users/User/Desktop/seminarska/cnn-text-classification-tf-master/cnn-text-classification-tf-master/data/rt-polarityFB.pos


Evaluating...

C:\Users\User\Desktop\seminarska\cnn-text-classification-tf-master\cnn-text-classification-tf-master\runs\1520282902\checkpoints\model-900
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "BestSplits" device_type: "CPU"') for unknown op: BestSplits
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "CountExtremelyRandomStats" device_type: "CPU"') for unknown op: CountExtremelyRandomStats
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "FinishedNodes" device_type: "CPU"') for unknown op: FinishedNodes
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "GrowTree" device_type: "CPU"') for unknown op: GrowTree
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "ReinterpretStringToFloat" device_type: "CPU"') for unknown op: ReinterpretStringToFloat
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "SampleInputs" device_type: "CPU"') for unknown op: SampleInputs
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "ScatterAddNdim" device_type: "CPU"') for unknown op: ScatterAddNdim
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "TopNInsert" device_type: "CPU"') for unknown op: TopNInsert
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "TopNRemove" device_type: "CPU"') for unknown op: TopNRemove
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "TreePredictions" device_type: "CPU"') for unknown op: TreePredictions
E d:\build\tensorflow\tensorflow-r1.0\tensorflow\core\framework\op_kernel.cc:943] OpKernel ('op: "UpdateFertileSlots" device_type: "CPU"') for unknown op: UpdateFertileSlots
Saving evaluation to C:/Users/User/Desktop/seminarska/cnn-text-classification-tf-master/cnn-text-classification-tf-master/runs/1520282902/checkpoints/..\prediction.csv
```

SLIKA 13: KLASIFIKACIJA KOMENTARJEV

Zaradi manjšega razlikovanja v strukturi učnih in testnih podatkov, ta metoda ni bila v veliko pomoč, saj s približno 75 odstotno natančnostjo sem moral, kljub vsemu ročno pregledati vse Facebook komentarje.

4.6 NEVRONSKA MREŽA V OBLAKU

Facebook komentarje sva popravljala dva pregledovalca in zaključila delo pri tisoč podatkovnih točkah (komentarjih). Vsi neprimerni komentarji so bili popravljeni, nepovezani označeni z oznako *Off topic* in komentarji v tujih jezikih s povedjo *Please write in English*.

Za rekurentno nevronsko mrežo za modeliranje sekvenc sem vzel že napisano kodo, priloženo učnemu članku, na uradni spletni strani Keras, katere knjižnica je tudi uporabljena.

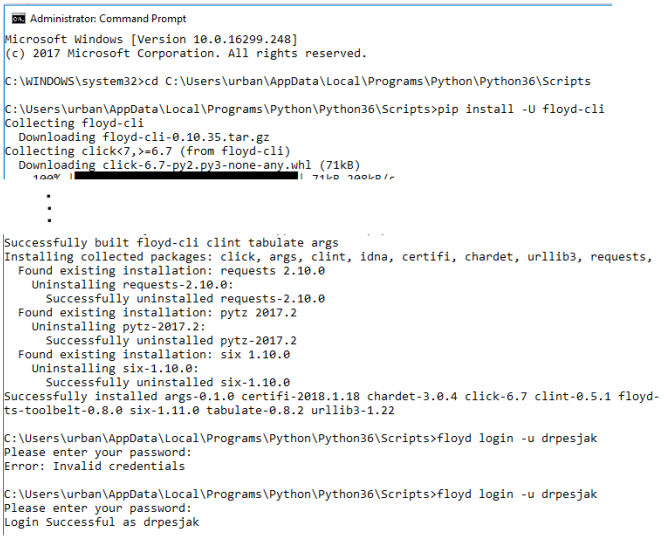
```
from __future__ import print_function

from keras.models import Model
from keras.layers import Input, LSTM, Dense
import numpy as np

batch_size = 64  # Batch size for training.
epochs = 100  # Number of epochs to train for.
latent_dim = 256  # Latent dimensionality of the encoding space.
num_samples = 10000  # Number of samples to train on.
# Path to the data txt file on disk.
data_path = 'fra-eng/fra.txt'

# Vectorize the data.
input_texts = []
target_texts = []
input_characters = set()
target_characters = set()
with open(data_path, 'r', encoding='utf-8') as f:
    lines = f.read().split('\n')
for line in lines[: min(num_samples, len(lines) - 1)]:
    input_text, target_text = line.split('\t')
    # We use "tab" as the "start sequence" character
    # for the targets, and "\n" as "end sequence" character.
    target_text = '\t' + target_text + '\n'
    input_texts.append(input_text)
    target_texts.append(target_text)
    for char in input_text:
        if char not in input_characters:
            input_characters.add(char)
    for char in target_text:
        if char not in target_characters:
            target_characters.add(char)

input_characters = sorted(list(input_characters))
target_characters = sorted(list(target_characters))
num_encoder_tokens = len(input_characters)
num_decoder_tokens = len(target_characters)
max_encoder_seq_length = max([len(txt) for txt in input_texts])
max_decoder_seq_length = max([len(txt) for txt in target_texts])

print('Number of samples:', len(input_texts))
print('Number of unique input tokens:', num_encoder_tokens)
print('Number of unique output tokens:', num_decoder_tokens)
print('Max sequence length for inputs:', max_encoder_seq_length)
print('Max sequence length for outputs:', max_decoder_seq_length)

input_token_index = dict(
    [(char, i) for i, char in enumerate(input_characters)])
target_token_index = dict(
    [(char, i) for i, char in enumerate(target_characters)])

encoder_input_data = np.zeros(
    (len(input_texts), max_encoder_seq_length, num_encoder_tokens),
    dtype='float32')
decoder_input_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens),
    dtype='float32')
decoder_target_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens),
    dtype='float32')

for i, (input_text, target_text) in enumerate(zip(input_texts, target_texts)):
    for t, char in enumerate(input_text):
        encoder_input_data[i, t, input_token_index[char]] = 1.
    for t, char in enumerate(target_text):
        # decoder_target_data is ahead of decoder_input_data by one timestep
        decoder_input_data[i, t, target_token_index[char]] = 1.
        if t > 0:
            # decoder_target_data will be ahead by one timestep
            # and will not include the start character.
            decoder_target_data[i, t - 1, target_token_index[char]] = 1.

# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs,
                                     initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)

# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)

# Run training
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
model.fit([encoder_input_data, decoder_input_data], decoder_target_data,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2)
# Save model
model.save('s2s.h5')

# Next: inference mode (sampling).
# Here's the drill:
# 1) encode input and retrieve initial decoder state
# 2) run one step of decoder with this initial state
# and a "start of sequence" token as target.
# Output will be the next target token
# 3) Repeat with the current target token and current states

# Define sampling models
encoder_model = Model(encoder_inputs, encoder_states)

decoder_state_input_h = Input(shape=(latent_dim,))
decoder_state_input_c = Input(shape=(latent_dim,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(
    decoder_inputs, initial_state=decoder_states_inputs)
decoder_states = [state_h, state_c]
decoder_outputs = decoder_dense(decoder_outputs)
decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_outputs] + decoder_states)

# Reverse-lookup token index to decode sequences back to
# something readable.
reverse_input_char_index = dict(

def decode_sequence(input_seq):
    # Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)

    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1, 1, num_decoder_tokens))
    # Populate the first character of target sequence with the start character.
    target_seq[0, 0, target_token_index['\t']] = 1.

    # Sampling loop for a batch of sequences
    # (to simplify, here we assume a batch of size 1).
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict(
            [target_seq] + states_value)

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = reverse_target_char_index[sampled_token_index]
        decoded_sentence += sampled_char

        # Exit condition: either hit max length
        # or find stop character.
        if (sampled_char == '\n' or
            len(decoded_sentence) > max_decoder_seq_length):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1, 1, num_decoder_tokens))
        target_seq[0, 0, sampled_token_index] = 1.

        # Update states
        states_value = [h, c]

    return decoded_sentence
```

SLIKA 14: PROGRAM REKURENTNE NEVRONSKE MREŽE ZA MODELIRANJE SEKVENC

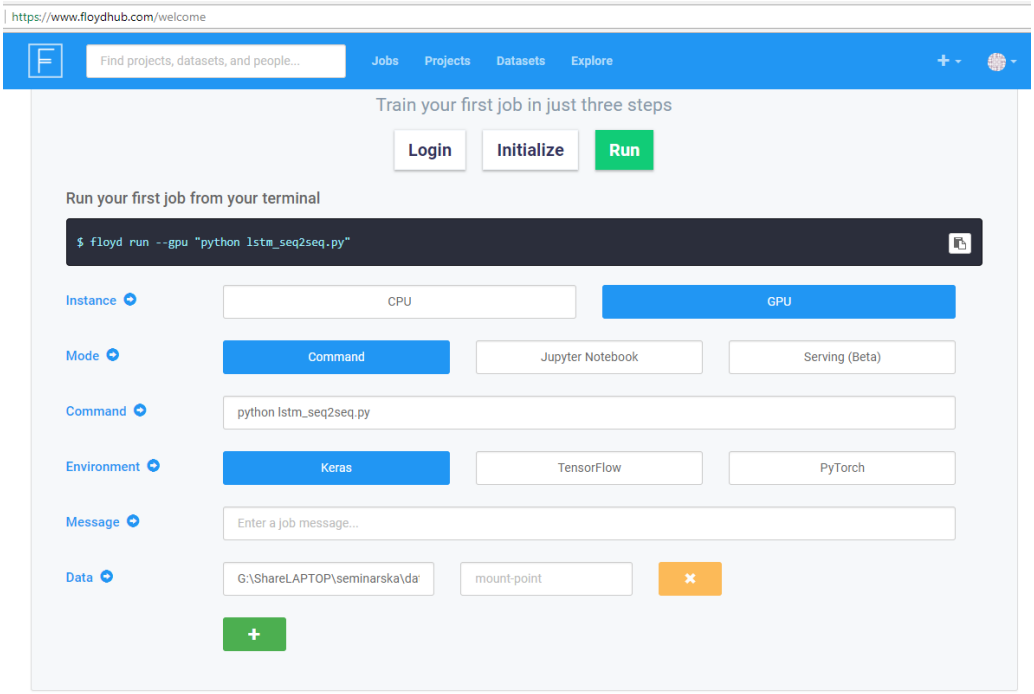
Glede na to da so nevronske mreže potratne v smislu procesorske moči, sem se odločil uporabiti oblačno storitev za procesiranje. Izbiral sem med Amazon Web Services, Google Cloud Platform in FloydHub. Medtem ko AWS in Google Cloud ponujata več načinov mikro optimizacije, je FloydHub uporabniku prijazneši in enostavnejši za uporabo. Poleg tega pa je uporabniku ob registraciji na voljo sto ur GPE uporabe.

Po registraciji na FloydHub-u sem s Python-ovim programom pip naložil vmesnik floyd in vse ostale potrebne knjižnice. Nato pa sem se z naslednji ukazom prijavil z uporabo uporabniškega gesla.



SLIKA 15: INSTALACIJA FLOYDHUB KLIENTA

Naslednji korak je bil le še zagon programa v oblaku z ukazom, ki sem ga generiral v grafičnem vmesniku oblačne storitve. Moral se le določiti katerem procesorju po program tekel, v kakšnem načinu, kateri ukaz želim izbrati in katero okolje. V ukazu sem zapisal ime programa, dodal pa sem še lokacijo podatkov.



SLIKA 16: FLOYDHUB GRAFIČNI VMESNIK

Generiran ukaz sem pognal v lokalni ukazni vrstici in kmalu dobil opomnik o končanem delu. Kot pričakovano se model nevronske mreže pretirano prilagaja učni množici, kar pomeni da ni sposoben posplošiti popravljanje komentarjev na še ne videne primere.

5 ZAKLJUČEK

5.1 POTRDITEV HIPOTEZ

1. Delo moderatorja je možno avtomatizirati.
2. Žaljive komentarje je možno izboljšati.
3. Vsebina članka izboljša natančnost modela strojnega učenja.
4. Z nevronsko mrežo lahko komentar naredimo prijazen.

Prvo hipotezo lahko potrdimo, saj so že mnogi pred menoj[21] to z modeli za detekcijo žaljivega govora storili, tudi z natančnostjo do 87 odstotkov, kar je dva odstotka več kot zastavljena ciljna meja za potrditev te hipoteze.

Potrdimo lahko tudi drugo hipotezo, z dejstvom da smo od pregledanih 1000 komentarjev, razrešili prav vsak primer, četudi še tako zahteven.

Model, zaradi relativno majhne količine podatkov v učni množici, slabo posplošuje, zato zadnji dve hipotezi ostajata nedefinirani.

5.2 NADALJNJE DELO IN PREDLOGI

V prihodnjih tednih nameravam nabor podatkov razširiti na vsaj 10-tisoč primerkov. Raziskavi bi lahko še dodali demografske značilnosti kot so spol. Odveč pa ne bi bilo niti večje število pregledovalcev oziroma začasnih moderatorjev, saj žaljiv, še posebej pa neprimeren govor, vsak posameznik tolerira do drugačne meje.

5.3 KAJ SEM SE NAUČIL

V preteklih nekaj mesecih sem podrobneje spoznal delovanje različnih nevronskih mrež in nekaterih algoritmov strojnega učenja, obnovil sem znanje v programskem jeziku Python in ga nadgradil z uporabo različnih knjižnic globokega učenja ter obdelavo big data. Seznanil sem se tudi z računalništvom v oblaku in delom podatkovnega analitika Vse ta znanja pa mi dajejo odlične temelje za mojo prihodnost na tem področju

6 VIRI

[1] *A Beginner's Guide to Recurrent Networks and LSTMs*. (brez datuma). Pridobljeno iz DL4J: https://deeplearning4j.org/lstm.html

[2] *A ten minute introduction to sequence to sequence learning in keras*. (brez datuma). Pridobljeno iz Keras blog: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

[3] Andreas Müller, H. B.-A. (brez datuma). *Machine learning with python*. Pridobljeno iz datacamp.com: https://www.datacamp.com/tracks/machine-learning-with-python

[4] Brownlee, J. (5. maj 2016). *Introduction python deep learning library tensorflow*. Pridobljeno iz Machine learning mastery: https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/

[5] *CNN for text - image.* (brez datuma). Pridobljeno iz Paddle paddle: http://paddlepaddle.org/docs/develop/book/05.recommender_system/index.html

[6] Colah. (27. avgust 2015). *Understanding LSTMs*. Pridobljeno iz http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[7] Davidson, T. (brez datuma). *Hate speech and offensive language*. Pridobljeno iz GitHub: https://github.com/t-davidson/hate-speech-and-offensive-language

[8] *Definition of AI*. (brez datuma). Pridobljeno iz Tech Target: http://searchcio.techtarget.com/definition/AI

[9] dennybritz. (brez datuma). *CNN for text classification*. Pridobljeno iz GitHub: https://github.com/dennybritz/cnn-text-classification-tf

[10] Glosser.ca. (28. 2 2013). *Colored neural network image.* Pridobljeno iz Wikipedia: https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

[11] Hovy, D. (Izvajalec). (maj 2017). *NLP meets Computational Social (Media) Science.* Pridobljeno iz http://videolectures.net/clarinplusworkshop2017_hovy_media_science/

[12] *Introduction to machine learning*. (brez datuma). Pridobljeno iz TUM Wiki: https://wiki.tum.de/display/lfdv/Introduction

[13] jbencina. (brez datuma). *Facebook news*. Pridobljeno iz GitHub: https://github.com/jbencina/facebook-news

[14] Kofler, M. (4. avgust 2017). *Deep learning with tensorflow*. Pridobljeno iz Towards data science: https://towardsdatascience.com/deep-learning-with-tensorflow-part-3-music-and-text-generation-8a3fbfdc5e9b

[15] *LSTML - seq2seq example*. (brez datuma). Pridobljeno iz GitHub: https://github.com/keras-team/keras/blob/master/examples/lstm_seq2seq.py

[16] Mesch, G. S. (brez datuma). *Family Relations and the Internet: Exploring a Family Boundaries Approach.* Pridobljeno iz https://doi.org/10.1207/s15327698jfc0602_2

[17] *Miscellaneous operating system interfaces*. (brez datuma). Pridobljeno iz Python documentation: https://docs.python.org/3.6/library/os.html

[18] Moody, C. (brez datuma). *word2vec.* Pridobljeno iz slideshare: https://www.slideshare.net/ChristopherMoody3/word2vec-lda-and-introducing-a-new-hybrid-algorithm-lda2vec-57135994

[19] *NumPy*. (brez datuma). Pridobljeno iz Wikipedia: https://en.wikipedia.org/wiki/NumPy

[20] *Pandas*. (brez datuma). Pridobljeno iz Wikipedia: https://en.wikipedia.org/wiki/Pandas_(software)

[21] Pinkesh Badjatiya, S. G. (brez datuma). *Deep Learning for Hate Speech Detection in Tweets.* Pridobljeno iz http://papers.www2017.com.au.s3-website-ap-southeast-2.amazonaws.com/companion/p759.pdf

[22] *Računalniki se učijo*. (22. 12 2015). Pridobljeno iz Monitor: http://www.monitor.si/clanek/racunalniki-se-ucijo/171558/

[23] Raval, S. (brez datuma). Cloud computing. Pridobljeno iz https://www.youtube.com/watch?v=Bgwujw-yom8

[24] Raval, S. (brez datuma). Intro to Deep learning. Pridobljeno iz https://www.youtube.com/watch?v=vOppzHpvTiQ&list=PL2-dafEMk2A7YdKv4XfKpfbTH5z6rEEj3

[25] Raval, S. (brez datuma). Learn Python for Data Science. Pridobljeno iz https://www.youtube.com/watch?v=T5pRlIbr6gg&list=PL2-dafEMk2A6QKz1mrk1uIGfHkC1zZ6UU

[26] Raval, S. (brez datuma). Machine learning for Heckers. Pridobljeno iz https://www.youtube.com/watch?v=2FOXR16mLow&list=PL2-dafEMk2A4ut2pyv0fSIXqOzXtBGkLj

[27] *Razlika med sovražnim, nesprejemljivim in neprimernim govorom*. (brez datuma). Pridobljeno iz Spletno oko: https://www.spletno-oko.si/faq/kaksna-je-razlika-med-sovraznim-nesprejemljivim-in-neprimernim-govorom

[28] *Regular expressions*. (brez datuma). Pridobljeno iz Python documentation: https://docs.python.org/3.6/library/re.html

[29] Robnik-Šikonja, d. M. (december 2017). *Globoke nevronske mreze za besedila.* Pridobljeno iz FRI - Univerza v Ljubljani: https://ucilnica.fri.uni-lj.si/pluginfile.php/41294/mod_folder/content/0/ONJ-11-Globoke%20nevronske%20mre%C5%BEe%20za%20besedila.pdf?forcedownload=1

[30] Samer Hijazi, R. K. (brez datuma). *Using Convolutional Neural Networks for Image Recognition.* Pridobljeno iz Cadence: https://ip.cadence.com/uploads/901/cnn_wp-pdf

[31] Schouwenaars, F. (brez datuma). *Intermediate python for data science*. Pridobljeno iz datacamp.com: https://www.datacamp.com/courses/intermediate-python-for-data-science

[32] Schouwenaars, F. (brez datuma). *Intro to python for data science*. Pridobljeno iz datacamp.com: https://www.datacamp.com/courses/intro-to-python-for-data-science

[33] Thomas Davidson, D. W. (brez datuma). *Automated Hate Speech Detection and the Problem of Offensive Language.* Pridobljeno iz AAAI: https://aaai.org/ocs/index.php/ICWSM/ICWSM17/paper/view/15665/14843

[34] ujjwalkarn. (11. 8 2016). *An Intuitive Explanation of Convolutional Neural Networks*. Pridobljeno iz ujjwalkarn.me: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[35] Urška Vranjek Ošlak, A. C. (brez datuma). *Prava frekvenca – analiza žaljivega govora v spletnih komentarjih.* Pridobljeno iz http://nl.ijs.si/janes/wp-content/uploads/2015/11/JANES15-013-Prava-frekvenca.pdf

[36] Zeerak Waseem, D. H. (brez datuma). *Hateful Symbols or Hateful People?* Pridobljeno iz http://www.aclweb.org/anthology/N16-2013

IZJAVA*

Mentor (-ica) , DUŠAN FUGINA, v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi naslovom
ALI JE LAHKO INTERNET PRIJAZEN,
katere avtorji (-ice) so DREJC PEŠJAK, _____, _____ :

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo (-ičino) dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje

Celje, 12.3.2018 žig šole Podpis mentorja(-ice)

Podpis odgovorne osebe

*
POJASNILO

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja(-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja(-ice) fotografskega gradiva, katerega ni avtor(-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.