TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4751

Primjena modela dubokog učenja za otkrivanje stavova u korisničkim komentarima

Bartol Freškura

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 11. ožujka 2016.

ZAVRŠNI ZADATAK br. 4751

Pristupnik: **Bartol Freškura (0036480392)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Primjena modela dubokog učenja za otkrivanje stavova u korisničkim komentarima**

Opis zadatka:

Duboko učenje naziv je za skup algoritama strojnoga učenja koji kroz višeslojnu obradu podataka modeliraju apstrakcije visoke razine. Ti su se modeli pokazali iznimno učinkoviti na nizu zadataka obrade prirodnog jezika, uključivo analizi sentimenta. Otkrivanje stavova novo je i izazovno područje u okviru analize sentimenta koje se bavi automatskom klasifikacijom i analizom stavova izraženih u tekstu, primjerice korisničkih komentara na internetu.
U okviru završnoga rada potrebno je upoznati se s osnovama neuronskih mreža, a zatim proučiti modele dubokog učenja temeljene na tradicionalnim povratnim neuronskim mrežama (eng. recurrent neural networks, RNN), modele dugog kratkoročnog pamćenja (engl. long short term memory, LSTM) te modele s propusnim povratnim jedinicama (gated recurrent units, GRU). Upoznati se s problemom analize stavova u korisnički generiranom sadržaju. Razviti i implementirati model za analizu sentimenta temeljen na dubokom učenju te ga primijeniti na prikladne skupove podataka na hrvatskome i engleskome jeziku. Provesti eksperimentalno vrednovanje modela, uključivo usporedbu s referentnim modelom i statističku obradu rezultata. Radu priložiti izvorni i izvršni kod razvijenog sustava, označene skupove podataka i potrebnu dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:

Doc. dr. sc. Jan Šnajder

Djelovođa:

Doc. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

Zahvaljujem tati, koji je imao strpljenja prevoditi svaku sliku u ovom radu i koji me je usmjerio prema računarstvu te uvijek bio spreman pomoći kada mi nešto nije bilo jasno.

Mami, koja je pročitala cijeli rad kako bi mi ukazala na pravopisne pogreške i koja je mi je uvijek kuhala najfinija jela i podržavala me u svemu što sam radio.

Mentoru Janu, koji mi je pružio priliku da budem dio TakeLaba i koji mi je pružao najbolje savjete kad su bili najviše potrebni. Također zahvale svim ostalim članovima TakeLaba na svoj pomoći koju su pružili, a posebno Martinu jer je uvijek imao vremena za mene i strpljivo je odgovarao na sva moja pitanja.

Svim ostalim članovima obitelji i prijateljima na podršci u mojim najgorim i najboljim trenutcima.

SADRŽAJ

1. Uvod

Analiza sentimenta jedna je od poznatijih tehnika koja pripada području obrade prirodnog jezika (engl. *natural language processing*), skraćeno, NLP. Cilj analize sentimenta je klasifikacija teksta u dokumentu ili rečenici s obzirom na to da li je izraženo mišljenje u njima pozitivno, negativno ili neutralno. Kao banalan primjer možemo uzeti dvije rečenice: *Danas je lijep dan.* i *Danas je ružan dan*. Prva rečenica izražava pozitivno mišljenje pa joj pridajemo pozitivnu oznaku, a drugoj rečenici negativnu oznaku jer ima negativan ton. Postoje i složenije vrste analize sentimenata u kojima postoji više razina oznaka, kao npr. pridavanje emocija proizvoljnom tekstu, no podaci u ovom radu imaju samo dvije razine oznaka, pozitivnu koja će se označavati s brojem jedan te negativnu koja će se označavati s brojem nula.

NLP je blisko vezan sa strojnim učenjem u kojem su razvijeni mnogi klasifikacijski algoritmi poput SVM-a i Naivnoga Bayesa koji pokušavaju klasificirati tekst pomoću matematičkih funkcija. Drukčiji pristup nude neuronske mreže, koje se koriste u razne svrhe strojnog učenja pa tako i za klasifikaciju teksta. Neuronske mreže su inspirirane biološkim neuronskim mrežama (centralni živčani sustav životinja, najčešće mozak) i koriste se za aproksimiranje matematičkih funkcija koje ovise o velikom broju ulaza.

Korak dalje ide područje *dubokog učenja*, koje također koristi neuronske mreže, no takve mreže imaju više od jednog skrivena sloja što im omogućava da nauče klasificirati mnogo složenije podatke s boljom preciznošću. Iako je teorija neuronskih mreža i dubokog učenja bila vrlo rano postavljena, najveći problem su predstavljali računalni resursi. Jedan od primjera je bio problem raspoznavanja poštanskih brojava na pošti. Algoritmi su davali dobre rezultate, no vrijeme potrebno za učenje tih mreža je bilo tri dana [12], što ih je činilo nepraktično za široku uporabu.

Početkom 2000-ih se vratio interes za korištenje dubokih neuronskih mreža upravo zbog razloga što se snaga računala razvijala eksponencijalnom brzinom. Veliko ubrzanje u učenju dubokih mreža je omogućilo i iskorištavanje resursa grafičkih kartica koje su u zadnjih nekoliko godina postale znatno moćnije od običnih računalnih procesora. Danas se duboke neuronske mreže koriste u mnogim problemima strojnog učenja kao

što su: prevođenje jezika (govor i tekst), prepoznavanje objekata sa slika, igranje igara itd.

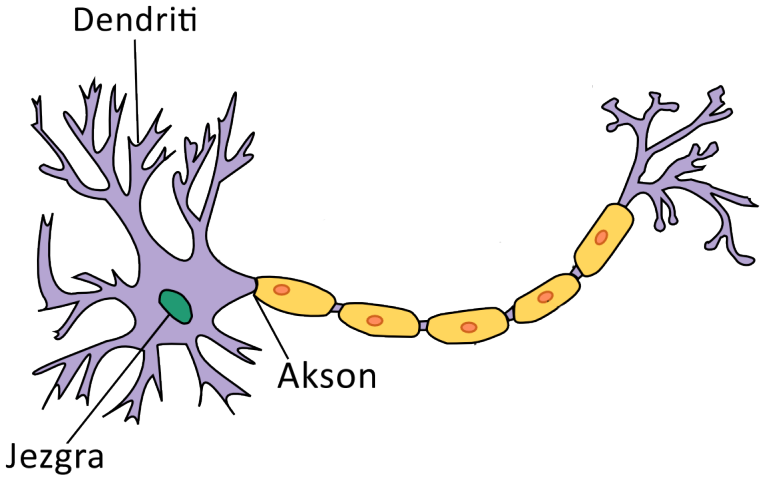
Cilj rada je analizirati performanse dubokih neuronskih mreža u problemu otkrivanja stavova u korisničkim komentarima. U radu su korištena dva modela dubokih mreža: LSTM (*long short term memory*) i GRU (*gated recurrent unit*). Oba modela mreže su vrlo popularna i pokazali su vrlo obećavajuće rezultate u problemu analize sentimenata [2]. Pomoću navedenih modela naučili smo duboku neuronsku mrežu raspoznavati pozitivan i negativan stav u komentarima korisnika.

U poglavlju 2 opisane su klasične neuronske i pojmovi koji su vezani uz njih, te strojno učeneje općenito. Slijedi opis dubokih mreža u poglavlju 3 u kojem se daje uvid u konfiguracije mreža koje se koriste u radu. Poglavlje 4 sadrži opis podataka koji su korišteni u eksperimentima te nakon toga prezentira postignute rezultate mreža s raznim paramaterima. Također se prikazuje usporedba s klasičnim algoritmom strojnog učenja.

2. Umjetne neuronske mreže

Umjetne neuronske mreže (engl. *artificial neural network*, dalje navođene kao neuronske mreže) su iz obitelji matematičkih modela koji su inspirirani biološkim neuronskim sustavima kao što su centralni živčani sustav u životinja (mozak osobito). Modeli se koriste za aproksimaciju matematičkih funkcija koje mogu ovisiti o velikom broju ulaza.

Mozgovi *Homo Sapiensa* napravljeni su od oko sto milijardi malih jedinica koje zovemo neuroni. Svaki od tih neurona je povezan s tisuću drugih neurona i komunicira s njima pomoću električnih signala. Princip funkcioniranja neurona je da nakon što primi signal iz dendrita[1] koji su povezani na njega, ako napon svih signala prijeđe određeni prag, taj neuron se "pali" i prosljeđuje svoj električni signal dalje drugim neuronima.



Slika 2.1: Dijelovi neuronske stanice [2]

Neuronske mreže funkcioniraju na vrlo sličan način. Također se sastoje od neurona, koji se kod neuronskih mreža zovu čvorovi (engl. *nodes*) i veza koje prenose in-

[1] Dendriti su veze s kojima je jezgra neurona povezana s ostalim neuronima.

[2] `http://users.tamuk.edu/kfjab02/Biology/AnimalPhysiology/B3408%20Systems/systems%20images/neuron.png.jpg`

formaciju od čvora do čvora. Umjesto električnih signala, ovdje se prenose numeričke vrijednosti ili najčešće matrice numeričkih vrijednosti koje predstavljaju informaciju. Ovakve vrste neuronskih mreža nazivamo acikličkim (engl. *feedforward*) mrežama.

2.1. Osnovna jedinica neuronske mreže

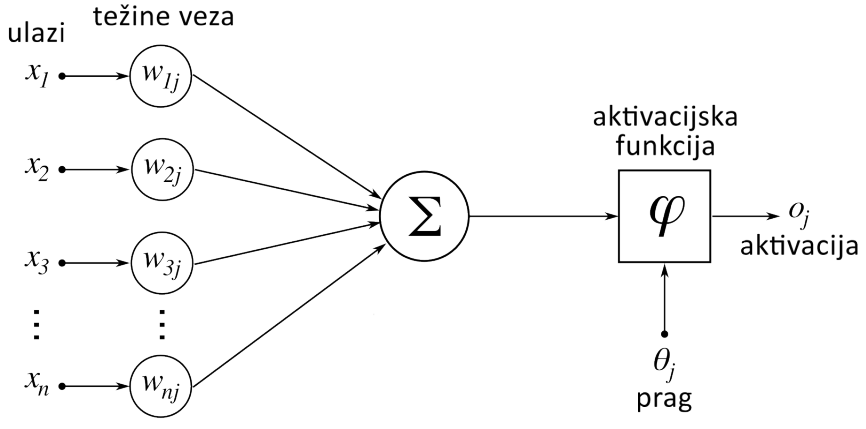
Jedan čvor u neuronskoj mreži može imati proizvoljan broj ulaza, a time i proizvoljan broj težina (engl. *weights*) koje su vezane na te ulaze. Sa n ćemo označavati broj ulaza u jedan čvor, a sa w_i težinu koja je vezana na i-ti ulaz, te sa x_i ulaz koji se množi s pripadajućom težinom. Izlaz iz čvora se prema tome računa po sljedećoj formuli:

$$\sum_{i=0}^{n} w_i x_i \tag{2.1}$$

Nakon što se izračuna izlaz, na njega se primjenjuje tzv. aktivacijska funkcija (engl. *activation function*) koja uz zadani prag određuje konačan izlaz iz čvora. Poznati primjer aktivacijske funkcije je sigmoida, no zbog loših svojstva (derivacija teži u nula na rubovima domene [16]) gubi važnost u primjeni:

$$\phi(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

$$h(x) = \phi(\sum_{i=0}^{n} w_i x_i) \tag{2.3}$$

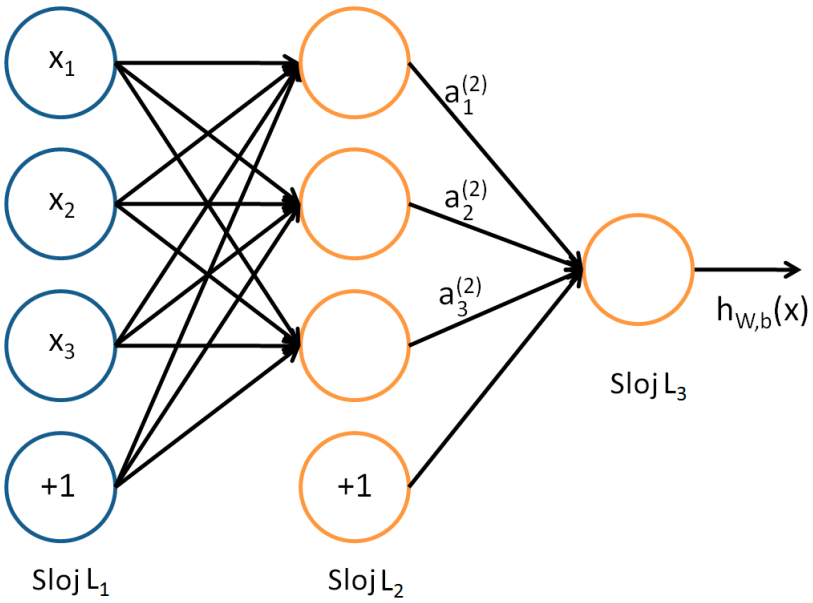


Slika 2.2: Dijelovi jednog čvora neuronske mreže[3]

[3] https://upload.wikimedia.org/wikipedia/commons/thumb/6/60/ArtificialNeuronModel_english.png/600px-ArtificialNeuronModel_english.png

2.2. Model neuronske mreže

Isto kako u živčanom sustavu postoji više od jednog neurona, tako i neuronska mreža ima mnogo čvorova [17].



Slika 2.3: Model višeslojne neuronske mreže[4]

Na slici 2.3 prikazana je mala neuronska mrežu koja se sastoji od tri sloja. Prvi sloj s lijeva se zove ulazni sloj (engl. *input layer*), slijedi jedan ili više skrivenih slojeva (engl. *hidden layer*) i na kraju se nalazi izlazni sloj (engl. *output layer*) koji može imati jedan ili više čvorova. Čvorovi označeni sa +1 se zovu pomaci (engl. *bias unit*) i one služe za pomicanje grafa aktivacijske funkcije.

Ako sa n označimo broj slojeva, sa L_l jedan sloj, sa $W_{i,j}^{(l)}$ težinu veze u sloju l, sa $b_i^{(l)}$ pristrani čvor sloja l, sa $a_i^{(l)}$ izlaz iz čvora, te s $h_{W,b}$ izlaz mreže, onda se može zapisati: [5]

$$a_1^{(2)} = \phi(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \tag{2.4}$$
$$a_2^{(2)} = \phi(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \tag{2.5}$$
$$a_3^{(2)} = \phi(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \tag{2.6}$$
$$h_{W,b}(x) = a_1^{(3)} = \phi(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}) \tag{2.7}$$

[4] `http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/`

[5] Češće se koristi matrični zapis i račun umjesto ovakvog zapisa.

Funkcija $\phi(x)$ kao i prije označava aktivacijsku funkciju. Radi kraće notacije, sa $z_i^{(l)}$ se označava ukupna suma ulaza i težina veza čvora i, u sloju l. Prema tome:

$$z_i^{(l+1)} = \sum_{j=1}^{n} W_{ij}^{(l)} x_j + b_i^{(l)} \tag{2.8}$$

Cijeli ovaj postupak u kojem želimo dobiti konačan izlaz iz neuronske mreže zove se unaprijedna propagacija (engl. *forward propagation*).

2.3. Algoritam propagacije unazad

Algoritam propagacije unazad (engl. *backpropagation algorithm*) [18] je algoritam koji omogućava osvježavanje težina veza u svim slojevima. Recimo da postoji problem u kojem se nalazi skup ulaza u mrežu $\left\{x^{(1)}, \cdots, x^{(m)}\right\}$ te izlaze iz mreže $\left\{y^{(1)}, \cdots, y^{(m)}\right\}$. Sada se može definirati funkcija gubitka (engl. *loss function*) J, koja će reći koliko dobro mreža predviđa izlaz i ta će mjera služiti za mjerenje performansi mreže:[6]

$$J(W, b; x, y) = \frac{1}{2} \left\| h_{W,b}(x) - y \right\|^2 \tag{2.9}$$

Treba napomenuti da se ovaj račun radi nad skupom podataka za učenje (engl. *training set*). Ideja je da mreža prilikom učenja vidi samo podatke iz skupa za učenje te se nakon učenja testira na dosad neviđenom skupu podataka kojeg zovemo testni skup (engl. *test set*). Na ovaj način se mjere stvarne performanse mreže, jer mreža nema velike koristi ako dobro ocjenjuje samo podatke koje joj eksplicitno damo. Cilj je postići da mreža radi dobro s bilo kojim podacima.

Konačna formula funkcije gubitka je:

$$J(W, b) = \left[\frac{1}{m} \sum_{i=1}^{m} J(W, b; x^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l - 1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \tag{2.10}$$

$$= \left[\frac{1}{m} \sum_{i=1}^{m} \left(\frac{1}{2} \left\| h_{W,b}(x^{(i)}) - y^{(i)} \right\|^2 \right) \right] + \frac{\lambda}{2} \sum_{l=1}^{n_l - 1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} \left(W_{ji}^{(l)} \right)^2 \tag{2.11}$$

Trostruke sume koje se pojavljuju označavaju regularizaciju koja služi smanjivanju utjecaja težina na vezama. U konačnici to onemogućuje pretreniranje (engl. *overfit*) mreže što znači da mreža točno računa izlaze samo za podatke pomoću kojih je učena.

[6] Ova funkcija gubitka se zove funkcija kvadratne greške.

Parametar λ se zove parametar raspada težina (engl. *weight decay parameter*). On određuje koliko će regularizacijski član imati utjecaja u konačnom rezultatu funkcije gubitka. Povećanje parametra može poboljšati sposobnost generalizacije neuronske mreže.

Gore navedena funkcija gubitka koristi se u klasifikacijskim i regresijskim problemima. Prilikom binarne klasifikacije postoji izlaz y koji poprima vrijednosti $\{0, 1\}$. Kako na izlazu neuronske mreže postoji sigmoidalna aktivacijska funkcija čija je slika na intervalu $[0, 1]$, sve vrijednosti izlaza $h_{W,b}(x) \geq 0.5$ se mogu označiti s brojem jedan, a $h_{W,b}(x) < 0.5$ s brojem nula.[7]

$$h'_{W,b}(x) = \begin{Bmatrix} 0, & za & h_{W,b}(x) < 0.5 \\ 1, & za & h_{W,b}(x) \geq 0.5 \end{Bmatrix} \tag{2.12}$$

Regresijski problemi na izlazu mreže nemaju podjelu prema klasama, već im je izlaz proizvoljan broj iz skupa realnih brojeva. Primjer regresije je neuronska mreža koja nastoji odrediti visinu čovjeka na temelju njegove težine.

Prije nego što se pokrene učenje neuronske mreže, moraju se postaviti težine veza i pristrani čvorovi na početne vrijednosti. Ne smiju se postaviti svi na nulu ili iste vrijednosti jer to znači da će svi skriveni čvorovi naučiti iste funkcije čime mreža postaje beskorisna. Najjednostavniji način je da sve inicijaliziramo prema normalnoj distribuciji s parametrima $\mu = 0$, $\sigma = 0.01$.

Učenje se svodi na mijenjanje težina veza i vrijednosti pristranih čvorova. Jedan od osnovnih algoritama te namjene je algoritam gradijentnog spusta. Algoritam gradijentnog spusta (engl. *gradient descent*) je opisan na sljedeći način:

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{2.13}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{2.14}$$

Intuicija algoritma propagacije unazad je sljedeća. Nakon što se izračunaju sve aktivacije u skrivenim slojevima te se dobije izlaz iz mreže $h_{W,b}(x)$, računa se greška $\delta_i^{(l)}$ za sve čvorove u mreži. Član greške $\delta_i^{(l)}$ zapravo mjeri koliko je svaki čvor "odgovoran" za konačnu grešku na izlazu mreže. Za zadnji čvor je jednostavno izračunati grešku pomoću 2.9. Greška skrivenih čvorova se računa tako da tražimo težinski prosjek svih članova greške čvorova koji koriste $a_i^{(l)}$ kao ulaz.
Konačno, algoritam nazadne propagacije je sljedeći:

[7] Ovaj pristup je korišten u radu.

1. Izračunaju se aktivacije za čvorove u skrivenim slojevima te za čvorove u izlaznom sloju.

2. Za svaki čvor i u sloju n_l (izlazni sloj):

$$\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)}) \tag{2.15}$$

3. Za $l = n_l - 1, n_l - 2, n_l - 3, \ldots, 2$:
   za svaki čvor i u sloju l:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)}) \tag{2.16}$$

4. Izračunaj parcijalne derivacije:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{2.17}$$

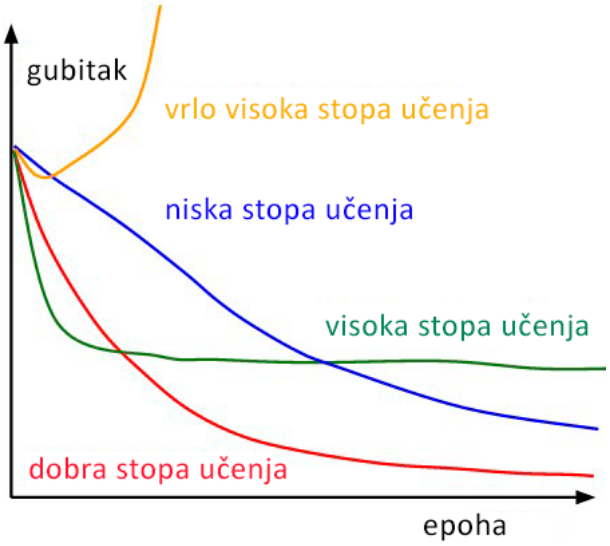
$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}. \tag{2.18}$$

2.4. Parametri neuronskih mreža

Nakon izgradnje neuronske mreže slijedi podešavanje parametara koji u konačnici utječu na performanse mreže. Osnovni parametri neuronskih mreža su broj skrivenih slojeva i dimenzije ulaza i izlaza. Osnovni parametri mijenjanju strukturu mreže, ali postoje i parametri koji ne mijenjaju strukturu, već mijenjaju intrinzična svojstva mreže.

2.4.1. Stopa učenja

Stopa učenja (engl. *learning rate*) je parametar kojem samo ime govori da određuje koliko brzo ili sporo mreža uči parametre.



Slika 2.4: Performanse učenja prilikom korištenja različitih stopa učenja [8]

Slika 2.4 prikazuje utjecaj stope učenja na performanse mreže. Ako se postavi previsoko, mreža neće moći konvergirati u globalni minimum[9], već će se zaustaviti u jednom od lokalnih minimuma ili će čak divergirati. Obrnuta situacija nastaje ako se stopa postavi jako nisko. Mreža će onda konvergirati u globalni minimum, no za to će joj trebati mnogo više vremena što je isto neželjen efekt. Odabirom ispravne stope učenja, mreža će konvergirati u globalni minimum u minimalno potrebnom vremenu.

2.4.2. Veličina uzorka i epohe

Veličina uzorka (engl. *batch size*) je pojam koji se veže uz računanje derivacije funkcije gubitka po težinama veza. Ako se prilikom računanja derivacija uzimaju svi primjeri iz skupa za treniranje, onda oni predstavljaju jedan uzorak čija je veličina određena upravo veličinom skupa za treniranje. Problem s ovim pristupom se očituje ako je skup za treniranje jako velik. To znači da se za jednu iteraciju osvježavanja parametara mora čekati da se izračunaju derivacije u odnosu na sve primjere, što može potrajati. Uz problem veličine se povlači i problem prilagođavanja stope učenja koja se onda mora prilagoditi za cijeli skup podataka. Zbog toga se uvode mini uzorci (engl. *mini batch*) koji nasumično podijele skup za učenje na više manjih podskupova. Zatim se računa srednja vrijednost greške podskupa te se računaju derivacije nad tim podskupovima. U

[8] `https://cs231n.github.io/assets/nn3/learningrates.jpeg`
[9] Minimum se odnosi na minimum funkcije gubitka.

formuli 2.19 k označava broj između jedan i m i on određuje veličinu mini uzorka.

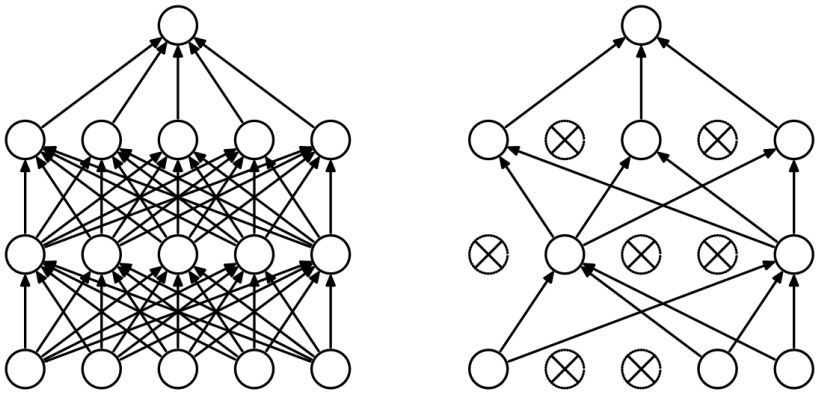
$$J(W,b) = \frac{1}{k}\sum_{i=1}^{k} J(W,b;x^{(i)},y^{(i)}) \tag{2.19}$$

Prednost ovog pristupa leži u činjenici da se parametri računaju mnogo češće, čime se omogućuje brža konvergencija neuronske mreže.

Ako se koristi pristup mini uzorcima, onda se odmah definira i pojam epohe. Epoha označava jedan prolazak kroz sve mini uzorke. U radu je korišten nešto viši broj epoha (50) nego uobičajno (oko 20), jer je broj primjera bio relativno mali.

2.4.3. Postotak ispadanja

Postotak ispadanja ili samo ispadanje [20] (engl. *dropout ratio* ili *dropout*) je tehnika kojom se gasi određen postotak nasumičnih veza ili čvorova u neuronskoj mreži.



Slika 2.5: Prikaz mreže bez ispadanja (slika *(a)*) i s ispadanjem (slika *(b)*) [10]

Kao posljedica ispadanja poboljšana je sposobnost generalizacije neuronske mreže, jer veze nasumično gube informaciju o susjednim vezama (manje ovise o njihovim vrijednostima) što znači da će više ovisiti same o svojim težinama.

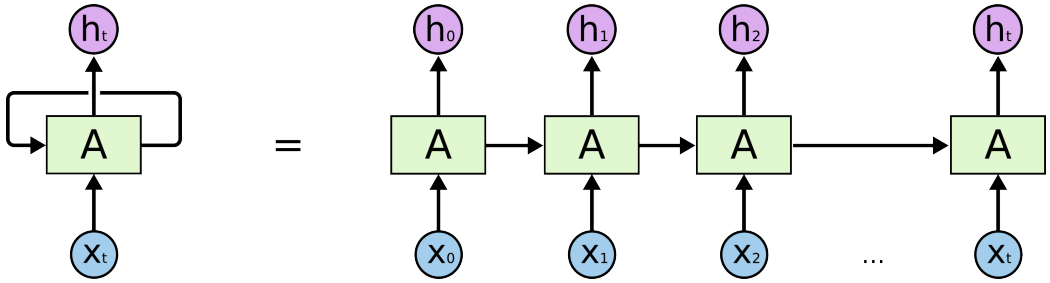
[10] `https://everglory99.github.io/Intro_DL_TCC/intro_dl_images/dropout1.png`

3. Duboko učenje

Duboko učenje (engl. *deep learning*) je relativno nova grana strojnog učenja koja se temelji na učenju različitih reprezentacija podataka. Razlika između učenja pomoću klasičnih neuronskih mreža i dubokog učenja je u broju skrivenih slojeva, čiji veći broj omogućava bolju reprezentaciju podataka. Postoji više vrsta dubokih neuronskih mreža, kao što su konvolucijske duboke mreže (engl. *convolutional deep network*), duboke probabilističke mreže (engl. *deep belief network*) i povratne neuronske mreže (engl. *recurrent neural network*) koje se koriste u ovom radu. Duboke mreže danas imaju mnogo primjena poput računalnog vida, prepoznavanja govora, obrade prirodnog jezika i bioinformatike te u tim područjima ostvaruju iznimne rezultate.

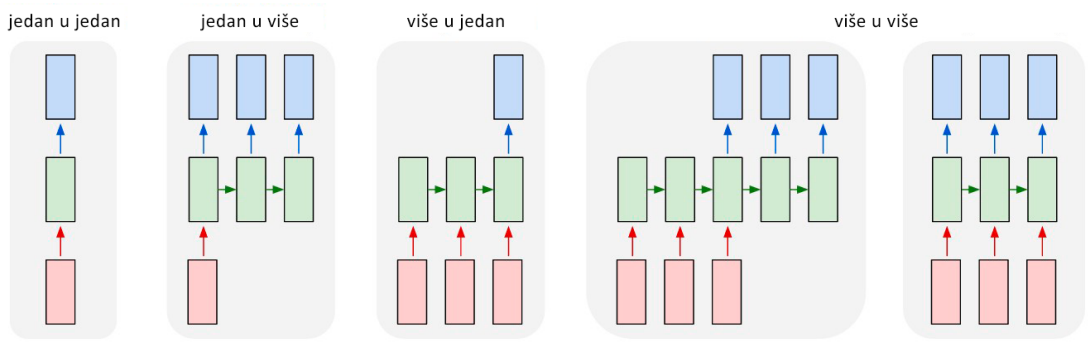
3.1. Povratne neuronske mreže

Ideja povratnih neuronskih veza (engl. *recurrent neural networks*) [14] (dalje kao RNN) je da se iskoristi slijed informacija. U tradicionalnim neuronskim mrežama se pretpostavlja da su svi ulazi i izlazi neovisni jedni o drugima. Iako takva pretpostavka u nekim vrstama problema daje dobre rezultate, u mnogim drugim problemima nam to nije dovoljno. Jedan od takvih problema je generiranje smislenog teksta. Ako želimo predvidjeti sljedeću riječ, onda moramo znati koje su se riječi već pojavile u tekstu. RNN-ovi se zovu tako upravo zbog tih sposobnosti. Svaki čvor ima dva ulaza: jedan iz prethodnog sloja te jedan povratni ulaz koji je zaslužan za čuvanje informacije o prethodnim ulazima.



Slika 3.1: Razmotan čvor RNN-a[1]

Primjer ulaza u RNN je rečenica koja se sastoji primjerice od pet riječi. To znači da će se mreža razmotati u neuronsku mrežu s pet slojeva, gdje će svaka pojedina riječ biti jedan ulaz u tu mrežu. Broj razmotavanja se u praksi naziva veličina vremenskog koraka (engl. *timestep*). Još jedno svojstvo koje čini RNN zanimljivim je mogućnost postojanja slijeda izlaza.



Slika 3.2: Razne kombinacije broja ulaza i izlaza u RNN-u[2]

Na slici 3.2 je prikazano pet različitih mogućnosti za ulaze i izlaze. Prva slika s lijeva prikazuje najjednostavniji oblik RNN-a gdje je fiksiran broj ulaza i izlaza. Problem klasifikacije slika se može opisati takvom mrežom. Na drugoj slici je složeniji slučaj u kojem se za jedan ulaz dobiva više izlaza. Ako se mreži na ulaz da slika, mreža može generirati opis te slike. Treća slika prikazuje obrnut slučaj koji se može opisati primjerom u kojem je ulaz rečenica, a izlaz je vrijednost sentimenta rečenice.[3] Četvrta slika je opis mreže koji se koristi za prevođenje rečenica s jednog jezika na drugi. Rečenica na hrvatskom jeziku je ulaz u mrežu, a izlaz iz mreže su riječi te rečenice na engleskom jeziku. Zadnja slika opisuje sinkronizirani slijed ulaza i izlaza. Primjer

[1] `https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/RNN-unrolled.png`

[2] `https://karpathy.github.io/assets/rnn/diags.jpeg`

[3] Ova vrsta mreže je korištena u radu.

korištenja je klasifikacija videa u kojem je cilj klasificirati svaki okvir videa. Treba napomenuti da je broj ulaza i izlaza RNN-a proizvoljan.

3.1.1. Matematički opis RNN-a

Kako su duboke neuronske mreže samo nastavak klasičnih neuronskih mreža, tako je i matematički opis prilično sličan. U RNN-u se pojavljuje nekoliko novih parametara, no ideja je i dalje ista. Ulazi se množe s težinama veza, na rezultat se primjenjuje aktivacijska funkcija te se postupak ponavlja kroz proizvoljan broj skrivenih slojeva, sve dok se ne dođe do izlaznog sloja. Na izlaz iz mreže se primjenjuje funkcija gubitka 2.9 te se gubitak vraća nazad kroz sve čvorove postupkom propagacije unazad. Prilikom propagacije unazad se osvježavaju vrijednosti težina na vezama te se na kraju dobije vrijednost derivacije funkcije gubitka po težinama za svaki čvor.

Ulazi će se označiti sa x_t, gdje je t broj vremenskog koraka. Ulazi mogu biti skalari ili vektori, no vektori su češći u primjeni. Primjer je vrijednost koja je kodirana vektorom koji ima samo jednu jedinicu u sebi (engl. *one-hot encoding*). Ako je na raspolaganju skup podataka koji su npr. komentari filmova, riječi iz tog skupa podataka definiraju vokabular problema, koji u računalu možemo zapisati pomoću tablice raspršenog adresiranja (engl. *hash map*). Kodiranje jednom jedinicom će svakoj riječi u skupu podataka pridružiti vektor dimenzija N, gdje je N broj jedinstvenih riječi u našem skupu podataka, tj. veličina prije navedene mape. Sve pozicije u vektorima, osim jedne, će biti postavljene na nulu. Preostala pozicija će biti postavljena na jedan i ta pozicija će odgovarati indeksu riječi u tablici raspršenog adresiranja. Jednadžba 3.1.1 prikazuje kodiranje na rečenici koja se sastoji od tri riječi. Rečenica je dio skupa podataka u kojem postoji N jedinstvenih riječi.

$$x_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad x_3 = \left.\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}\right\} N \tag{3.1}$$

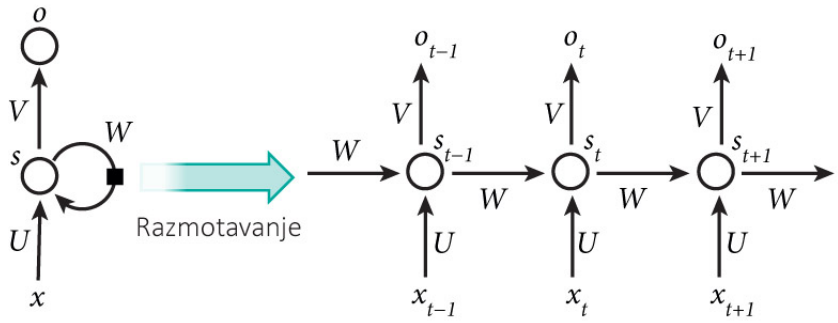
Na slici 3.3, član s_t označava stanje skrivene ćelije[4] u vremenskom koraku t. Član

[4]Ćelija je samo drugi naziv za čvor koji se primjenjuje kada se govori o RNN-u.

s_t je posljedica ulaza x_t i stanja ćelije u prethodnom vremenskom koraku s_{t-1}:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{3.2}$$

Član s_t predstavlja dosad skupljenu informaciju i može ga se promatrati kao memoriju mreže. On sadrži informacije o svim prethodnim ćelijama.



Slika 3.3: Detaljan prikaz jednog vremenskog sloja RNN-a[5]

Funkcija f je u neuronskim mrežama bila sigmoida, no u dubokim se mrežama koristi ili tangens hiperbolni ili ReLU (engl. *Rectified Linear Unit*).

Matrice U i V predstavljaju parametre jednog sloja. Njihov pandan u neuronskim mrežama su bile težine veza. Bitno je napomenuti da su te matrice jednake unutar jednog razvijenog vremenskog sloja t. To se može objasniti na način da se odrađuje isti posao u svakom vremenskom koraku, ali taj posao za svaki vremenski korak ima različit ulaz. Izlaz iz ćelije u vremenskom koraku t se označava sa o_t. Izlaz se množi s matricom V koja je isto dio parametara. Na taj umnožak se također primjenjuju funkcije ovisno o primjeni mreže. Jedna od takvih funkcija je *softmax* koja bi u primjeru s vokabularom davala vektor čiji su članovi vjerojatnosti riječi koje se mogu pojaviti na tom izlazu.[6]

$$o_t = softmax(Vs_t) \tag{3.3}$$

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{h} e^{x_j}} \tag{3.4}$$

Slika 3.3 prikazuje izlaze iz svake od ćelija, no nije nužno slučaj da svaka ćelija ima izlaz. Moguće je imati broj izlaza $i \in [1, m]$, gdje je m broj vremenskih koraka.

[5] `https://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/09/rnn.jpg`

[6] h označava broj mogućih klasa.

Nula naravno nije uključena jer bi to značilo da postoji sloj mreže koji nije povezan sa sljedećim slojem.

3.1.2. Propagacija unazad kroz vrijeme

Konačan cilj RNN-a je isti kao i u klasičnim neuronskim mrežama: minimizirati vrijednost koju vrati funkcija gubitka. Do vrijednosti funkcije gubitka se dolazi unaprijednom propagacijom isto kao i kod neuronskih mreža. Jedina razlika je u tome što RNN to radi kroz sve ćelije u svim vremenskim slojevima. Nakon izračuna gubitka potrebno je propagirati grešku natrag kroz mrežu i osvježiti matrice U, V i W postupkom propagacije unazad. Postupak propagacije unazad se u RNN-ovima naziva postupkom propagacije unazad kroz vrijeme (engl. *Backpropagation through time*, skraćeno BPTT) zbog vremenskih slojeva kroz koje se obavlja propagacija.

Izlaz iz mreže će se preimenovati iz o u $\hat{y}$ zbog konzistentnosti s ostalom literaturom. U neuronskim mrežama se na izlaz iz mreže primjenjivala sigmoidalna funkcija sa gubitkom srednjeg kvadratnog odstupanja, no ovdje će se to promijeniti na gubitak unakrsne entropije (engl. *cross entropy*) On je definiran kao:

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t \tag{3.5}$$

$$E(y, \hat{y}) = \sum_{t=0}^{m} E_t(y_t, \hat{y}_t) \tag{3.6}$$

$$= -\sum_{t=0}^{m} y_t \log \hat{y}_t \tag{3.7}$$

U klasičnim neuronskim mrežama postojala je samo jedna matrica (matrica težina veza) za koju je trebalo izračunati derivaciju u ovisnosti o funkciji gubitka. Ovdje se nalaze čak tri matrice za koje je potrebno učiniti istu stvar, kako bi se mogle osvježiti vrijednosti tih matrica pomoću postupka gradijentnog spusta. Za matricu V postupak je nešto jednostavniji jer se matrica ne pojavljuje u izračunu vremenskog koraka. Derivacije matrica W i U morat će uzeti u obzir i vremenski korak, zbog čega se izračun komplicira. Zbog vremenskog se koraka moraju zbrojiti svi gradijenti u jednom vremenskom sloju kako bi se dobio konačan gradijent za matrice W i U:

$$\frac{\partial E}{\partial W} = \sum_{t=0}^{m} \frac{\partial E_t}{\partial W} \tag{3.8}$$

$$\frac{\partial E}{\partial U} = \sum_{t=0}^{m} \frac{\partial E_t}{\partial U} \tag{3.9}$$

Nakon provedenog postupka nazadne propagacije, vrijednosti $\frac{\partial E}{\partial U}, \frac{\partial E}{\partial V}, \frac{\partial E}{\partial W}$ će biti izračunate.

Ulančano pravilo deriviranja se koristi prilikom deriviranja kompozicija funkcija. Ako postoje diferencijabilne funkcije $f(x)$ i $g(x)$, te funkcija $F(x) = (f \circ g)(x)$, onda vrijedi $F'(x) = f'(g(x))g'(x)$. Ako se definiraju $y = f(u)$ i $u = g(x)$, onda $\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$ čime se objašnjava prije navedena derivacija funkcije $F(x)$. Prvo će se pokazati izračun za $\frac{\partial E_t}{\partial V}$, koji je najjednostavniji. Slijedeći ulančano pravilo, derivacija po matrici V se može zapisati na ovaj način:[7]

$$\frac{\partial E_t}{\partial V} = \frac{\partial E_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial V} \tag{3.10}$$

$$= \left(-\sum_k y_k \frac{\partial log(\hat{y}_k)}{\partial z_t}\right) \otimes s_t = \left(-\sum_k y_k \frac{1}{\hat{y}_k}\frac{\partial \hat{y}_t}{z_i}\right) \otimes s_t \tag{3.11}$$

$$= \left(-y_t(1-\hat{y}_t) - \sum_{k\neq t} y_k \frac{1}{\hat{y}_k}(-\hat{y}_k\hat{y}_t)\right) \otimes s_t \tag{3.12}$$

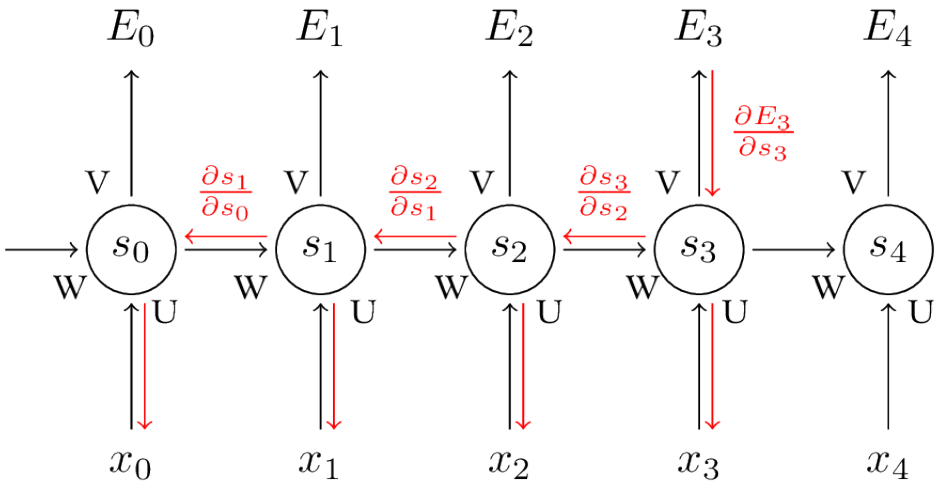
$$= \left(-y_t(1-\hat{y}_t) + \sum_{k\neq t} y_k(\hat{y}_t)\right) \otimes s_t \tag{3.13}$$

$$= \left(-y_t + y_t\hat{y}_t + \sum_{k\neq t} y_k\hat{y}_t)\right) \otimes s_t \tag{3.14}$$

$$= \left(\hat{y}_t \sum_k y_k - y_t\right) \otimes s_t \tag{3.15}$$

$$= (\hat{y}_t - y_t) \otimes s_t \tag{3.16}$$

[7] $\otimes$ označava matrično množenje prvog vektora s transponiranim drugim vektorom.



Slika 3.4: Rekurzivne ovisnosti ulaza s u algoritmu BPTT za vremenski korak $t = 3$[8]

Sada je vidljivo da je $\frac{\partial E_t}{\partial V}$ ovisi samo o $\hat{y}_t, y_t, s_t$, i moguće ga je izračunati jednostavnim oduzimanjem i matričnim množenjem.

Za $\frac{\partial E_t}{\partial W}$ se stvari kompliciraju zbog sumacije u vremenskom koraku (isto vrijedi i za $\frac{\partial E_t}{\partial U}$). Može se zapisati sljedeće:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial W} \tag{3.17}$$

U ovom primjeru se koristi tangens hiperbolni za aktivacijsku funkciju, stoga vrijedi $s_t = \tanh(Ux_t + Ws_{t-1})$. Sa slike 3.4 se vidi da s_t ovisi o s_{t-1} rekurzivno, sve do s_0 ($s_1 = \tanh(Ux_1 + Ws_0)$). Zbog toga se mora koristiti ulančano pravilo za izračun $\frac{\partial s_k + 1}{\partial s_k}$, sve dok se ne dođe do $k = 0$.

$$\frac{\partial s_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \cdots \frac{\partial s_1}{\partial s_0} \frac{\partial s_k}{\partial W} \tag{3.18}$$

Uvrštavanjem jednadžbe 3.18 u 3.17 dobiva se sljedeće:

$$\frac{\partial E_t}{\partial W} = \sum_{k=0}^{t} \frac{\partial E_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \left(\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W} \tag{3.19}$$

Suma 3.19 opisuje zbrajanje greške kroz vremenski sloj i daje ukupnu grešku u vremenskom sloju. Nakon što je to izračunato, primjenjuje se gradijentni spust kao i u

[8] `http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/10/rnn-bptt-with-gradients.png`

neuronskim mrežama (2.14) kako bi se dobila promjena vrijednosti u matricama W i U.

3.1.3. Problem nestajućeg gradijenta

Problem nestajućeg gradijenta (engl. *vanishing gradient problem*) je svojstvo RNN-ova zbog kojeg oni loše funkcioniraju kada je riječ o dužim ovisnostima (veliki vremenski koraci). U primjeni je to problem jer kod dužih rečenica možda baš prva ili druga riječ rečenice određuje smisao rečenice: *Moj pas, koji me uvijek dočeka spreman da ide van kada se vratim s posla, danas me nije dočekao i to me je malo zabrinulo.* RNN bi ovdje naučio da je netko zabrinut, no ne bi mogao znati tko i zbog čega, jer su subjekt i objekt rečenice navedeni na početku rečenice.

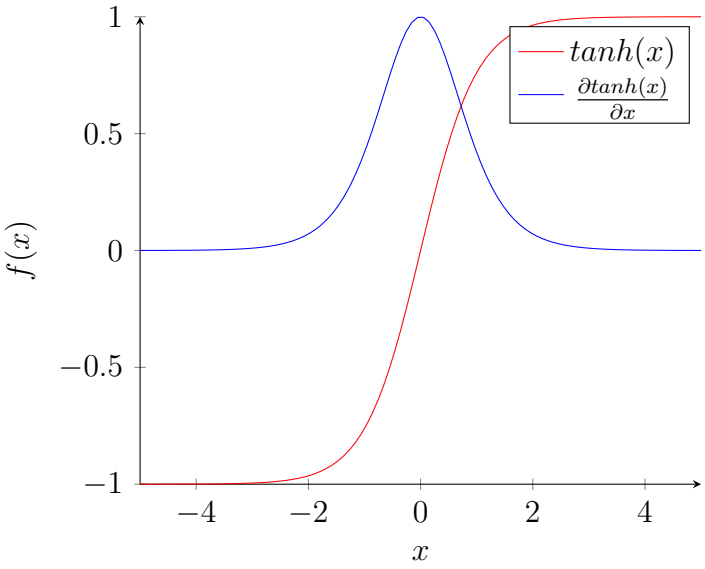
Ako se pogleda 3.19, vidi se da se jedan član računa rekurzivno koristeći pravilo ulančane derivacije:

$$\prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}} \tag{3.20}$$

Gledajući graf funkcije $tanh(x)$ (3.5), koja se koristi kao aktivacijska funkcija, može se vidjeti da derivacija funkcije teži u nulu na oba kraja. To efektivno znači da će se gradijenti prijašnjih vremenskih slojeva svesti na vrlo male brojeve ako vrijednosti u parametarskim matricama poprime male vrijednosti (zbog mnogo operacija množenja vrijednosti se eksponencijalno smanjuju), čime više neće doprinositi učenju u višim vremenskim slojevima. U konačnici to vodi ka gubitku informacije u duljim nizovima riječi.

Postoji još i problem eksplodirajućeg gradijenta (engl. *exploding gradient problem*) [16] koji je upravo obrnut nestajućem gradijentu. Ako se početne vrijednosti parametarskih matrica postave na veće brojeve i ako je vremenski korak velik, u izrazu 3.19 te će se matrice množiti međusobno m puta u najgorem slučaju, što zbog mnogostrukog množenja može vrlo brzo dati NaN vrijednost u računalu. Nije nužno da matrice inicijalno budu postavljene na visoke vrijednosti. Tijekom treniranja je također moguće dobiti visoke vrijednosti u tim matricama što može rezultirati prevelikim brojem za računalo.

Aktivacijska funkcija ReLU nema takve probleme i zato se u praksi najčešće koristi. Drugo rješenje pruža duga kratkoročna memorija, skraćeno LSTM.

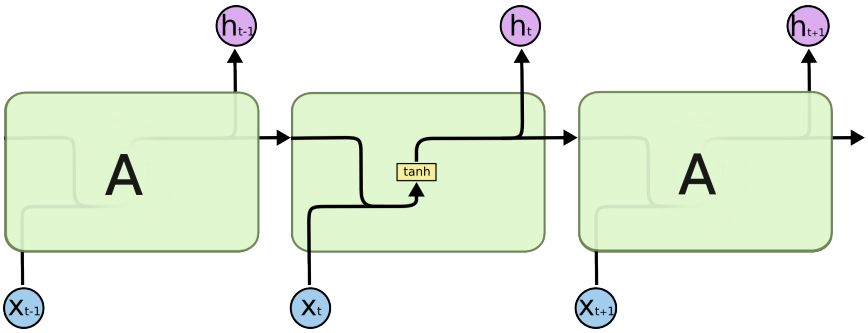


Slika 3.5: Graf funkcije $tanh$ i njene prve derivacije

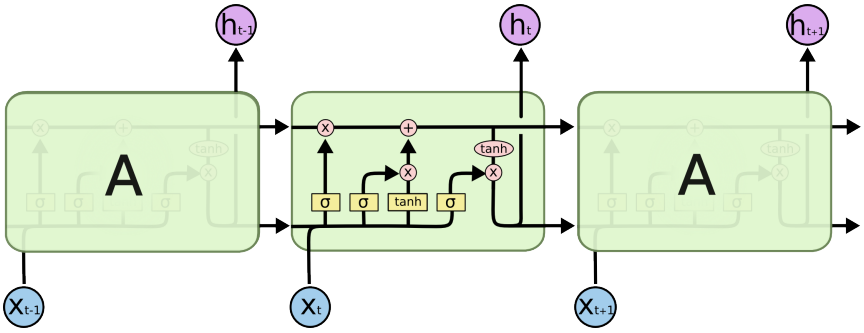
3.2. Duga kratkoročna memorija

Duga kratkoročna memorija (engl. *Long short term memory*, skraćeno LSTM) je posebna vrsta RNN-a koja rješava nedostatke RNN-a vezane uz probleme gradijenata i pamćenja informacije kroz više vremenskih koraka. Zbog tog svojstva su se pokazale kao dobro rješenje i danas se koriste u mnogim problemima. Prvu takvu mrežu su osmislili Hochreiter i Schmidhuber [9] i eksplicitno su je dizajnirali za rješavanje problema dugačkih ovisnosti u vremenskom koraku.

Ako se usporede dijelovi ćelija u RNN-u i LSTM-u (slike 3.6 i 3.7), vidi se da je razlika u dodatnom ulazu i izlazu te funkcijama koje se primjenjuju unutar same ćelije.

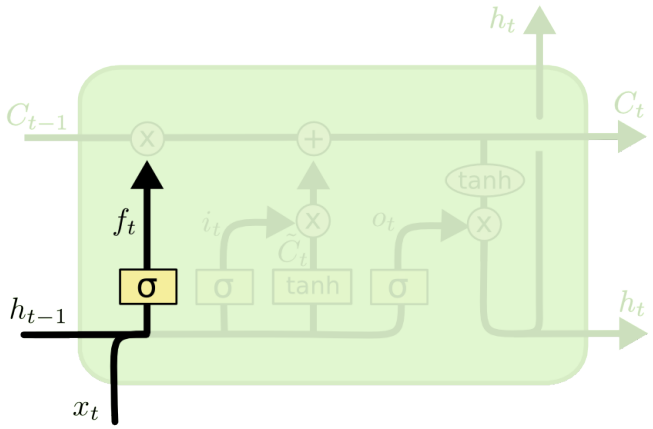


Slika 3.6: Ćelija RNN-a[9]



Slika 3.7: Ćelija LSTM-a[10]

Žute kućice označavaju naučene slojeve mreže, ružičasti krugovi operacije koje se obavljaju nad pojedinim elementima, linije koje se spajaju konkatenaciju, [11] a linije koje se razdvajaju označavaju dupliciranje. U nastavku su objašnjeni detalji svakog dijela ćelije.



Slika 3.8: Sloj zaborava[12]

Prva žuta kućica sa sigmoidalnom funkcijom se u LSTM-u naziva sloj zaborava (engl. *forget gate layer*). Intuitivno, on odlučuje koliko stare informacije će se očuvati prilikom računanja informacije za trenutnu ćeliju. Ako je vrijednost sigmoidalne funkcije nula, to znači da se stara informacija uopće ne uzima u obzir u trenutnoj ćeliji, dok izlaz jedan znači da će se cijela prijašnja informacija u potpunosti uzeti u obzir. (3.8)

[9] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-SimpleRNN.png

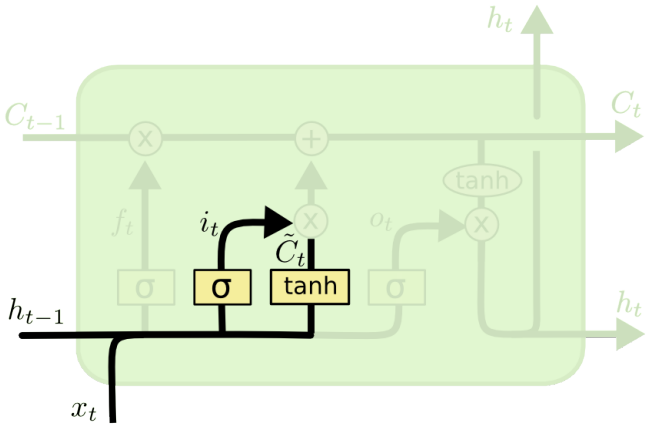
[10] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png

[11] U jednadžbama je to označeno pomoću uglatih zagrada.

[12] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-f.png

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.21}$$

Idući korak se odnosi na novu informaciju koja će se spremiti u ćeliju. Sloj sa sigmoidalnom funkcijom odlučuje koje nove vrijednosti treba osvježiti. Tangens hiperbolni sloj stvara nove potencijalne kandidate za trenutnu ćeliju. (3.9)



Slika 3.9: Funkcije koje odlučuju dio nove informacije zadržane u ćeliji[13]

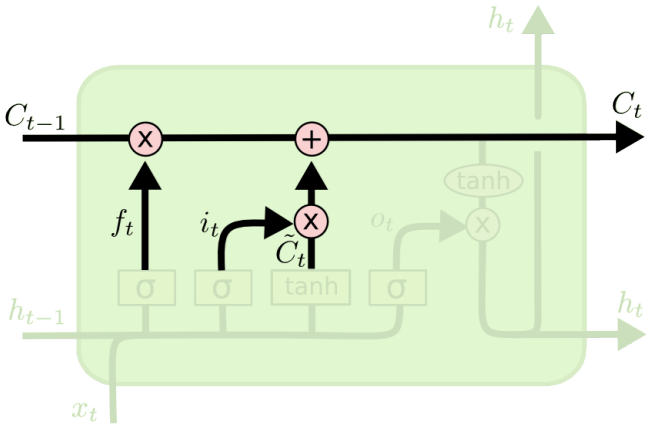
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.22}$$

$$\widetilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{3.23}$$

Sada se osvježava stanje ćelije zbrajajući staro stanje sa izabranim novim stanjem (3.10).

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{3.24}$$

[13] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-i.png

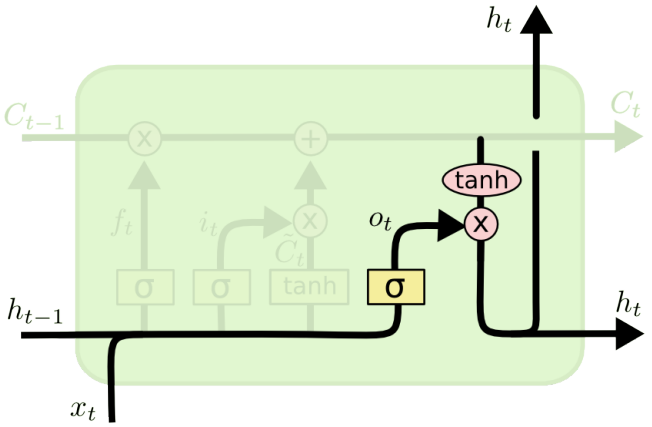


Slika 3.10: Osvježavanje stanja ćelije[14]

Konačno, računaju se vrijednosti koje se šalju višim slojevima i sljedećem vremenskom koraku. Opet se koristi sigmoidalna funkcija kako bi se filtrirala trenutna informacija te se množi s novoizračunatim stanjem. Prije množenja, na vrijednosti novog stanja se djeluje tangensom hiperbolnim kako bi se ograničile vrijednosti na interval $[-1, 1]$. (3.11)

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.25}$$

$$h_t = o_t * tanh(C_t) \tag{3.26}$$



Slika 3.11: Izračun konačnog izlaza iz ćelije[15]

[14] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-C.png

[15] https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-o.png

3.3. Propusna povratna ćelija

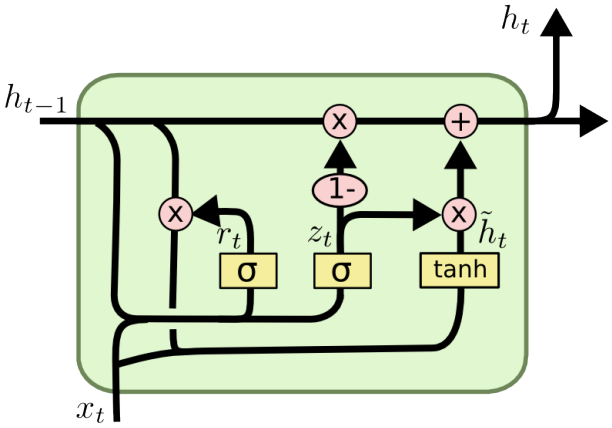
Propusna povratna ćelija (engl. *gated recurrent unit*, skraćeno GRU) [1] je varijanta LSTM-a koja kombinira sloj zaborava i ulazni sloj u jedan sloj koji se zove sloj osvježavanja. Također spaja stanje ćelije sa skrivenim stanjem te radi manje promjene u ostatku ćelije.

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{3.27}$$

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{3.28}$$

$$\widetilde{h}_t = tanh(W[r_t * h_{t-1}, x_t]) \tag{3.29}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t \tag{3.30}$$



Slika 3.12: Dijelovi GRU ćelije[16]

3.4. Optimizacijski algoritmi

Prilikom opisivanja dubokih i klasičnih neuronskih mreža, korištena je metoda gradijentnog spusta kao optimizacijski algoritam. Optimizacijski algoritmi općenito opisuju način osvježavanja težina veza u neuronskoj mreži. Stohastički gradijentni spust (engl. *stochastic gradient descent*)[17] jedan je od osnovnih optimizacijskih algoritama koji se koristi u strojnom učenju i opisan je u 2.14.

[16] `https://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-var-GRU.png`

[17] Isto kao običan gradijentni spust, samo se koristi mini uzorak umjesto cijelog skupa za učenje.

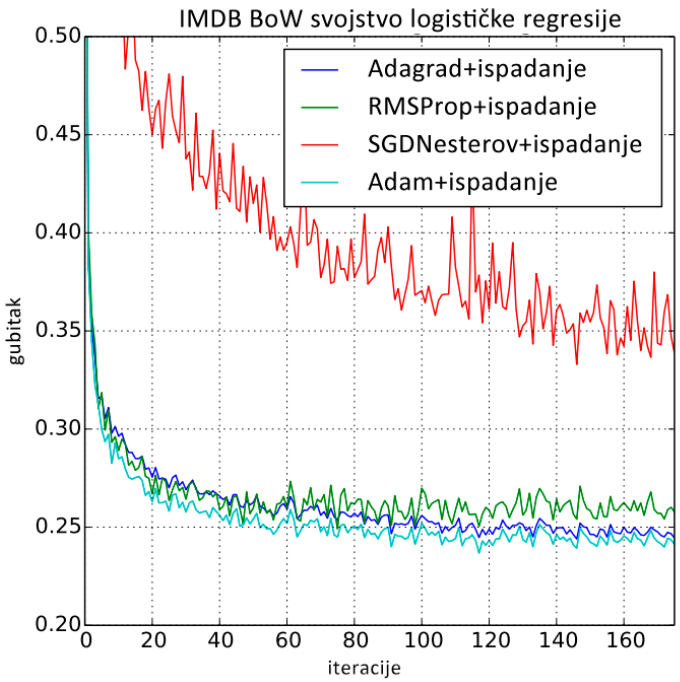
Tijekom vremena su se počele pojavljivati varijacije stohastičkog gradijentnog spusta koje rješavaju problem statičke stope učenja α i time sporog učenja. Gradijentni spust sa momentumom je nadogradnja koja omogućava modelu bržu konvergenciju na dijelovima gdje su derivacije funkcije gubitka jako blage (najčešće u području lokalnih minimuma).

$$v_{ij} = \gamma v_{ij} + \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{3.31}$$

$$W_{ij}^{(l)} = W_{ij}^{(l)} - v_{ij} \tag{3.32}$$

Član v se zove vektor brzine (engl. *velocity vector*) i iste je dimenzije kao parametar W. Parametar $\gamma \in (0, 1]$ određuje udio prošlih iteracija gradijenata dodanih u trenutno osvježavanje gradijenta. U početku učenja se stavlja na vrijednost oko 0.5, a nakon što se učenje stabilizira, polako se povećava na oko 0.9 više.

Postoji još mnogo optmizacijskih algoritama no među najboljima razlike postaju vrlo malene, tako da odabir algoritma ne igra veliku ulogu. Na slici 3.13 mogu se vidjeti performanse nekih od poznatijih optimizacijskih algoritama koji se koriste u praksi.



Slika 3.13: Brzine konvergiranja raznih optimizacijskih algoritama [11]

U radu je korišten *Adam* [11] algoritam. Ideja *Adam*-a je da kombinira prednosti

dva dosad najpopularnija algoritma: *AdaGrad* [5], koji radi dobro s rijetkim gradijentima, i *RMSProp*-om [23], koji radi dobro u učenju nad cijelim skupom podataka i s dinamičkim postavkama.
Algoritam *Adam* je sljedeći:

Input: α: Veličina koraka
Input: $\beta_1, \beta_2 \in [0, 1)$: Eksponencijalni raspad za očekivanja momenata
Input: $f(\theta)$: Stohastička funkcija gubitka s parametrom θ
Input: θ_0: Početni vektor parametara
$m_0 \leftarrow 0$: Inicijaliziraj prvi vektor momenta
$v_0 \leftarrow 0$: Inicijaliziraj drugi vektor momenta
$t \leftarrow 0$: Inicijaliziraj vremenski korak
while θ_t *nije konvergirao* **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Izračunaj derivaciju težina u odnosu na funkciju gubitka u vremenskom koraku t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Osvježi očekivanje prvog momenta)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Osvježi očekivanje drugog momenta)
  $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ (Izračunaj ispravljeno očekivanje prvog momenta)
  $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ (Izračunaj ispravljeno očekivanje drugog momenta)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ (Osvježi težine veza)
end
Output: θ_t

Algoritam 1: Optimizacijski algoritam *Adam*

4. Eksperimenti

Cilj rada je bio odrediti stav korisničkim komentara iz raznih izvora pomoću modela dubokih neuronskih mreža. U ovom poglavlju se eksperimentira s dva modela povratnih neuronskih mreža: LSTM i GRU. Mijenjanjem njhovih parametra se promatra utjecaj na razne skupove podataka. Eksperimentiranje se provodilo nad dva međusobno neovisna skupa podataka. Prvi skup podataka je na engleskom jeziku i sastoji se od mišljenja ispitanika o četiri različite debate. Skup podataka na hrvatskom jeziku se sastoji od korisničkih komentara s hrvatske *web*-stranice za naručivanje hrane.

4.1. Analiza stavova

U obradi prirodnog jezika, analiza stavova (engl. *stance analysis*) jest postupak automatskog utvrđivanja stava autora teksta prema nekoj zadanoj temi. Određivanje stavova temeljeno je na osvrtima i povratnoj informaciji korisnika koja proizlazi iz korisničkog iskustva. Zbog brzog razvoja tehnologije, većina ljudi danas obavlja poslove preko Interneta (engl. *online*). Registracije, rezervacije, bankarstvo i kupovanje samo su neke od poznatih usluga koje se danas obavljaju preko Interneta. Ovo omogućuje korisnicima da vrlo brzo ostavljaju svoja mišljenja u vezi usluga koje koriste u obliku komentara. Mišljenja korisnika mogu se klasificirati kao mišljenje za (engl. *favor*), protiv (engl. *against*) ili neutralno (engl. *neutral*). Pružatelji usluga stoga imaju iznimnu korist od mišljenja korisnika njihove usluge, što u glavnu ulogu postavlja analizu stavova kao primarni alat koji im pomaže u ekstrakciji mišljenja. Iz rezultata analize pružatelji usluga mogu uvidjeti zadovoljstvo ili nezadovoljstvo korisnika te poduzeti određene akcije s ciljem poboljšavanja usluge ako se ispostavi da je ocjena usluge loša.

Današnji pristup prema analizi stavova može se grupirati u tri grupe. Prva grupa su tehnike temeljene na bazi znanji (engl. *knowledge based*) koje pokušavaju klasificirati tekst prema riječima koje imaju jednu moguću interpretaciju (npr. sretan, tužan, dosadan) [21]. U drugoj grupi se nalaze statističke metode koje se oslanjaju na elemente

strojnog učenja kao što su SVM-ovi [1] i modeli vreće riječi. Naprednije metode pokušavaju odrediti vlasnika stava (onaj koji izražava svoje mišljenje) i metu stava (na koga se mišljenje odnosi). Kako bi se dobio stav iz konteksta, koriste se gramatičke veze riječi koje predstavljaju svojstva analiziranog teksta [4]. Hibridni pristup je treća grupa koja uzima u obzir strojno učenje, ali i elemente reprezentacije znanja poput ontologije i semantičkih mreža kako bi se mogli detektirati stavovi koji nisu jako izraženi. Primjer može biti analiza koncepata koji eksplicitno ne sadrže relevantnu informaciju, no implicitno su povezani s drugim konceptima koji sadrže relevantnu informaciju [3].

4.2. Opis podataka

4.2.1. Podatci na engleskom jeziku

Skup podataka na engleskom jeziku se sastoji od četiri neovisne domene:

- Obama (kratica OBA)
- Prava homoseksualaca (kratica PRHO)
- Marihuana (kratica MAR)
- Pobačaj (kratica ABO)

Podatci su skupljeni kroz popularnu online platformu za debate *CreateDebate*[2], i dostupni su su na stranici: `http://www.hlt.utdallas.edu/~saidul/stance/stance.html` [8].

Ispitanici su pitani za mišljenje o temama te su njihovi odgovori u cijelosti zabilježeni. Uz svaki je odgovor dodana i ocjena stava ispitanika. Razine sentimenata su pozitivno (oznaka 1) i negativno (oznaka -1). Prilikom obrade su oznake promijenjene na 1 i 0 jer korišteni algoritmi koriste takav format zapisa.

Tablica 4.1: Statistika pojedinih domena

	ABO	PRHO	OBA	MAR
Ukupno odgovora	1741	1376	985	626
Pozitivno označeni odgovori [%]	54.9	63.4	53.9	69.5

U tablici 4.1 prikazana je kratka statistika svih korištenih podataka [8].

[1] Kratica od engl. *Support Vector Machines*.

[2] `http://www.createdebate.com`

Tablica 4.2: Primjeri odgovora ispitanika

Odgovor	Stav
Many of the aborted fetus may have also grown up to be rapists, serial murders,terrorists, armed robbers... but since they were only potentials and not actual... then there's nothing more to add.	Za (1)
it is wrong to live in a perspective of not producing children when overpopulation isn't met. Do you know what "overpopulation" means? It is "naturally wrong" to have an infertile body when you can't produce an offspring Why?	Protiv (0)

Podaci su filitrirani na način da je primijenjen program *Aspell*[3] koji ispravlja pravopisne pogreške. Eksperimenti su provedeni na lematiziranom i nelematiziranom skupu podataka te su rezultati kasnije uspoređeni.

Lematizacija (engl. *lemmatisation*) je postupak kojim se različiti oblici riječi pretvaraju u jedan zajednički oblik. Primjer mogu biti riječi *trčao, trčali, trčati, zatrčavanje*, koje bi se lematizacjiom zamijenile s jednom riječi: *trčanje*. Posljedica lematizacije je daleko pojednostavljen vokabular skupa podataka, čime se mogu poboljšati rezultati modela jer model ne mora učiti veze između svih oblika riječi. Lematizacija je provedena pomoću alata *spaCy*.[4]

4.2.2. Podatci na hrvatskome jeziku

Skup podataka o prikupljenom mišljenju iz hrvatskih korisničkih komentara [6] je skup podataka koji se sastoji od korisničkih komentara preuzetih sa stranice Pauza.hr[5]. Pauza.hr je *web*-stranica preko koje korisnici mogu naručivati hranu iz restorana diljem Zagreba i okolice. Također je omogućeno pisanje komentara o pojedinim restoranima te ocjenjivanje restorana na skali od nula do šest, s inkrementom od 0.5.

Tablica 4.3: Statistika hrvatskih podataka

Ukupan broj komentara	Pozitivno označeni komentari [%]
3310	66.3

[3] `http://aspell.net/`
[4] `https://spacy.io/`
[5] `http://www.pauza.hr/`

Ocjene korisnika su izmijenjene na način da se dobije binarna klasifikacija. Sve ocjene veće od 4.5 su označene kao pozitivne, dok je sve manje ili jednako 4.5 označeno negativno. Nad korisničkim komentarima je provedena provjera pravopisa pomoću programa *Aspell*. Riječi su također lematizirane.

Tablica 4.4: Primjeri komentara i odgovarajućih klasifikacija sa *web*-stranice Pauza.hr

Komentar	Stav
pet pohvale za klopu, jedna od ukusnijih koju sam jeo iz dostave... osoblje jako ljubazno, samo tako nastavite i malo proucite kvartove po kojima dostavljate :-D	Za (1)
narucila sam pilece prutice u sezamu s pomfrijem i pilecu juhu. Nije losa hrana, za cetvorku, ali i meni su kasnili pola sata, zato snižavam ocjenu..	Protiv (0)

4.3. Parametri

Iako postoji mnogo parametara u dubokoj neuronskoj mreži, samo neki su dali znatniju promjenu u rezultatima. Veličina mini uzorka je postavljena na 64 zbog optimalnih performansi učenja.[6] Broj epoha je postavljen na 50 i nakon svake epohe se skup za učenje nasumično miješao kako mreža ne bi napamet naučila poredak podataka. Kao optimizacijski algoritam je korišten *Adam* jer se pokazao najboljim od svih poznatijih optimizacijskih algoritama Stopa učenja se nije pokazala kao presudan faktor te je davala vrlo slične rezultate za vrijednosti $[0.01, 0.0001]$. Kao konačna vrijednost je uzeta 0.001 i ta se vrijednost koristila se kroz sve eksperimente.

Parametri koji su mijenjani kroz eksperimente:

- Broj vremenskih koraka
- Broj skrivenih slojeva
- Dimenzija skrivenog sloja (dimenzija parametarskih matrica u skrivenom sloju)
- Postotak ispadanja
- Vrsta ćelije (LSTM ili GRU)

[6] `https://svail.github.io/rnn_perf/`

4.4. Eksperimenti nad engleskim podatcima

U engleskom skupu podataka su bile dostupne četiri domene koje su tretirane nezavisno jedna od druge. Razlog tome je što su tematike domena vrlo različite jedna od druge što povlači činjenicu da su argumenti, načini izražavanja i vrste riječi upotrebljavane u odgovorima, također različiti. Iako čovjek može s lakoćom odrediti stav autora u bilo kojoj od korištenih domena, dubokoj neuronskoj mreži je to iznimno teško, a osobito ako nema dovoljno velik uzorak podataka.

Umjesto kodiranja s jednom jedinicom, korišten je Googleov *word2vec*[7] model. Kodiranje jednom jedinicom svakoj riječi pridodaje jednu vrijednost (jedinicu u vektoru) što modelu ne predstavlja nikakvu informaciju o mogućim vezama između riječi. Ako je model prilikom učenja vidio riječ *mačka*, on zbog gore navedene reprezentacije riječi, neće moći zaključiti ništa ako mu se nakon *mačke* pojavi riječ *pas* (oboje su sisavci, imaju četiri noge, mogu biti ljubimci, itd.).

Model *word2vec* predstavlja riječi kao numeričke vektore, gdje su semantički slične riječi postavljene u točke prostora koje se nalaze jedna blizu druge. Vektori se temelje na hipotezi distribucije[8] , koja kaže da riječi koje se pojavljuju u istim kontektstima, dijele semantičko značenje. Vektori sadrže informaciju o mnogo lingvističkih pravila pa se primjerce može prepoznati da je $v(kralj) - v(muskarac) + v(djevojka) = v(kraljica)$. Kako bi naučio reprezentacije riječi, *word2vec* treba ogroman skup podataka na kojem može naučiti veze među riječima. U eksperimentima je korišten model koji je treniran na *Google News* skupu podataka [15] (oko sto milijardi riječi). Svaka riječ je reprezentirana s tristo dimenzionalnim vektorom, a ukupno ima tri milijuna riječi.

Svaki skup podataka je podijeljen na dva dijela. Prvi dio je skup za treniranje i predstavlja 66% skupa podataka, a ostatak je dodijeljen skupu za ispitivanje.

Tablica 4.5: Podjela skupa na ispitni skup i skup za učenje

	ABO	PRHO	OBA	MAR
Broj primjera u ispitnom skupu	592	468	335	213
Broj primjera u skupu za učenje	1149	908	650	413

[7] `https://code.google.com/archive/p/word2vec/`

[8] `https://en.wikipedia.org/wiki/Distributional_semantics#Distributional_Hypothesis`

4.4.1. Rezultati s nelematiziranim podacima

U ovoj konfiguraciji se riječi u skupu podataka nisu lematizirale, ali je nad njima provedeno ispravljanje pravopisnih pogrešaka.

Korištene su sljedeće oznake u tablicama: broj vremenskih koraka (BVK), broj skrivenih slojeva (BSS), dimenzija skrivenog sloja (DSS) te ispadanje (D[%]). Vektori u drugom redu u određenim tablicama korespondiraju gore navedenim oznakama.

Tablica 4.6: Preciznosti na testnom skupu (LSTM ćelija, bez lematizacije)

	BVK, BSS, DSS, D[%]			
	[70, 1, 100, 0]	[70, 1, 500, 0]	[70, 2, 300, 0]	[70, 3, 300, 10]
Obama	52.8%	54.7%	**56.9%**	48.1%
Prava homoseksualaca	55.1%	52.7%	**58.3%**	55.1%
Marihuana	61.5%	59.4%	59.9%	**63.5%**
Pobačaj	**52.9%**	52.1%	**52.9%**	47.7%

U tablici 4.6 su mijenjane dimenzije skrivenog sloja i broj skrivenih slojeva. Pokazalo se da je optimalan broj skrivenih slojeva dva (s tri sloja preciznost je jedino bolja za domenu marihuane), a dimenzija skrivenog sloja nema utjecaja na preciznost.

Tablica 4.7: Utjecaj vremenskog koraka (LSTM ćelija, bez lematizacije)

	BVK, BSS, DSS, D[%]			
	[100, 2, 300, 0]	[150, 2, 300, 0]	[40, 2, 300, 0]	[55, 2, 300, 0]
Obama	53.4%	47.5%	**60.6%**	57.2%
Prava homoseksualaca	52.3%	54.0%	**55.8%**	55.1%
Marihuana	56.2%	**69.8%**	61.9%	57.9%
Pobačaj	52.2%	52.6%	**56.6%**	53.1%

U tablici 4.7 je promatran utjecaj vremenskog koraka na preciznost. Ispostavilo se da je vremenski korak od 40 optimalan za tri od četiri domene (domena marihuane je pokazala puno bolje rezultate za vremenski korak od 150).

Tablica 4.8: Uzimanje različitih frekvencija (LSTM ćelija, bez lematizacije)

	BVK=40, BSS=2, DSS=300, D=0%			
	FD=100%	FD=70%	FD=50%	FD=30%
Obama	56.6%	60.0%	57.1%	**61.5%**
Prava homoseksualaca	**58.9%**	51.3%	56.3%	55.4%
Marihuana	60.4%	**67.2%**	64.1%	66.1%
Pobačaj	52.9%	53.9%	**55.9%**	55.3%

Tablica 4.8 prikazuje rezultate s mijenjanim postotkom riječi koje su ostavljene u vokabularu (postotak je označen kraticom FD). Iz skupa podataka je stvoren vokabular te su riječi poredane po svojoj učestalosti. Zatim je primijenjena funkcija koja je izbacivala sve riječi koje nisu bile u prvih $x\%$. Rezultati variraju ovisno o domeni i teško je zaključiti koji je optimalan postotak riječi koje treba sačuvati, iako je za tri od četiri domene rezultat malo bolji ako se koristi FD između 30% i 70%.

4.4.2. Rezultati s lematiziranim podacima

Podaci su u sljedećim rezultatima bili podvrgnuti najprije ispravljanju pravopisnih pogrešaka, a zatim i lematizaciji. Korišteni su parametri koji su u prosjeku davali najbolje rezultate s nelematiziranim podacima.

Tablica 4.9: Rezultati sa lematizacijom podataka (LSTM ćelija)

BVK=40, BSK=2, DSS=300, D=0%, FD=50%	
Obama	56.3%
Prava homoseksualaca	55.3%
Marihuana	58.9%
Pobačaj	55.2%

Ako se usporede rezultati iz tablice 4.9 s prijašnjim rezultatima, vidi se da je lematizacija malo pogoršala rezultate ako se gleda prosjek od svih domena. Najveći pad preciznosti se očituje u domeni marihuane.

4.4.3. Predtrenirani model na skupu podataka IMDb

Koristeći sljedeći pristup, prije učenja nad domenama, mreža najprije uči nad alternativnim skupom podataka te na kraju učenja sprema datoteku u kojoj se nalaze infor-

macije koje je naučila. Nakon toga se pokreće učenje nad svim domenama, ali im se daje spremljena datoteka kao ulaz te mreža zapravo nastavlja učiti, ali nad drukčijim skupom podataka. Cilj ovakvog pristupa je poboljšavanje rezultata na način da mreža nauči vrijednosti parametara u mreži nad dovoljno velikim skupom podataka te onda nad svakom od domena uči pojedine detalje rečenica, tj. prilagođava dodatno parametre za svaki od skupa podataka. Za alternativni skup podataka korišten je Standfordov veliki skup recenzija filmova [13], koji se sastoji od 50000 visoko polarnih recenzija raznih filmova od strane korisnika web stranice IMDb.[9] Mijenjan je i promatran utjecaj ispadanja (engl. *dropout*).

Tablica 4.10: Rezultati s predtreniranim modelom i mijenjanim ispadanjem (LSTM ćelija, bez lematizacije)

	BVK=40, BSS=2, DSS=300, D=0%		
	D=0%	D=20%	D=40%
Obama	60.0%	**60.6%**	60.0%
Prava homoseksualaca	54.9%	**57.1%**	53.3%
Marihuana	63.5%	**66.1%**	61.5%
Pobačaj	51.0%	**54.5%**	52.7%

Gledajući tablicu 4.10 vidi se da se rezultati nisu poboljšali kada smo ubacili predtrenirani model. Što se tiče postotka ispadanja, 20%-tno ispadanje je dalo najbolje rezultate za sve domene i sudeći po razlici rezultata (u prosjeku 2.2% bolji rezultati nego bez ispadanja), igra bitnu ulogu. Jedan od razloga zašto dropout poboljšava rezultate može biti sprječavanje pretreniranja. Izbacivanje nasumičnih skrivenih ćelija u skrivenim slojevima sprječava da mreža napamet nauči poredak riječi u skupu za učenje, što onda poboljšava sposobnost generalizacije i time rezultate u ispitnom skupu.

4.4.4. Rezultati s ćelijom GRU

Umjesto LSTM ćelije, primijenjena je GRU ćelija, a rezultati su prikazani u tablici 4.11 Obama je jedina tema u kojoj su se rezultati poboljšali, dok su ostalim domenama ostali približno isti kao i s LSTM ćelijom.

[9] `http://www.imdb.com/`

Tablica 4.11: Rezultati bez lematizacije podataka (GRU ćelija)

BVK=40, BSK=2, DSS=300, D=40%, FD=50%	
Obama	60.9%
Prava homoseksualaca	59.5%
Marihuana	66.1%
Pobačaj	53.8%

4.5. Eksperimenti nad hrvatskim podacima

Svi podaci iz hrvatskog skupa podataka su lematizirani i kao takvi se koriste u eksperimentima. Za kodiranje podataka se koristila tehnika kodiranja jednom jedinicom.

Tablica 4.12: Podjela skupa na ispitni skup i skup za treniranje

Broj primjera u ispitnom skupu	Broj primjera u skupu za učenje
1092	2218

Tablica 4.13: Preciznosti na ispitnom skupu (LSTM ćelija)

BVK, BSS, DSS, D[%]		
[30, 1, 300, 0]	[30, 2, 300, 0]	[30, 3, 300, 10]
65.9%	77.0%	**79.9%**

Tablica 4.13 pokazuje utjecaj broja skrivenih slojeva na rezultate. Iako u engleskim podacima tri sloja nisu poboljšala rezultate, ovdje je slučaj obrnut. Tri sloja poboljšavaju rezultat za čak 3%.

Tablica 4.14: Utjecaj vremenskog koraka (LSTM ćelija)

BVK, BSS, DSS, D[%]			
[50, 2, 300, 0]	[100, 2, 300, 0]	[20, 2, 300, 0]	[70, 2, 300, 0]
81.7%	70.7%	79.3%	80.5%

Vremenski korak od 50 se pokazao optimalan za ovaj skup podataka. Razlog tome je što su komentari uglavnom puno kraći od rasprava u debatama te je ovdje manji vremenski korak bolje reprezentirao informaciju komentara.

Tablica 4.15: Uzimanje različitih frekvencija (LSTM ćelija)

BVK=50, BSK=2, DSS=300, D=0%			
FD=100%	FD=70%	FD=50%	FD=30%
78.9%	79.0%	**82.0%**	74.5%

Izbacivanje neučestalih riječi je poboljšalo rezultate za par postotaka kao što se može vidjeti iz tablice 4.15.

Tablica 4.16: Utjecaj ispadanja (LSTM ćelija)

BVK=50, BSK=2, DSS=800, FD=50%		
D=0%	D=20%	D=40%
82.6%	**83.5%**	83.4%

Prilikom mijenjanja ispadanja (tablica 4.16) je promijenjena dimenzija skrivenog sloja s 300 na 800 jer se pokazalo da daje 2-3% bolje rezultate. Utjecaj ispadanja je relativno malen te se najbolji rezultati dobivaju ako se postavi na 20% ili 40%.

Tablica 4.17: Rezultati s GRU ćelijom

BVK=50, BSK=2, DSS=800, D=20%, FD=50%
81.9%

Mijenjanje ćelije u GRU ćeliju nije poboljšalo rezultate (tablica 4.17). Korišteni su parametri koji su davali optimalne rezultate kod LSTM ćelije.

4.6. Usporedba s klasičnim algoritmom strojnog učenja

Ovdje se uspoređuju rezultati dubokog učenja s najboljim rezultatima koji se dobivaju koristeći algoritam SVM. Sve postavke nad podatcima su ostale iste te se samo promijenio algoritam učenja i način kodiranja podataka. Za kodiranje je korišten model "vreće riječi" (engl. *bag of words*) s TF-IDF težinama riječi. Težine TF-IDF (engl. *term frequency – inverse document frequency*) je statistika koja je namijenjena određivanju važnosti riječi u dokumentu koji se nalazi u skupu više dokumenata. Nakon što se izračunaju statistike za svaku riječ u svakom dokumentu, te statistike se koriste u modelu vreće riječi umjesto diskretnih vrijednosti koje su uobičajne za model vreće riječi.

Parametri SVM-a su odabrani korištenjem pretraživanja rešetkom (engl. *grid search*) i odabrani su oni koji su dali najbolje rezultate.[10] Pretraživanje rešetkom je odrađeno na način da se izabralo mnogo različitih parametara za SVM te se skup za učenje podijelio u sedam jednakih dijelova. Zatim je model iterirao kroz svaki od sedam dijelova te mu je uvijek trenutni podskup bio ispitni skup, dok su svi ostali (šest njih) bili korišteni kao skupovi za učenje. Ova tehnika se zove *K-fold cross validation*. Nakon što procedura završi, uzme se kombinacija parametara koja je davala najbolje rezultate te se ti parametri iskoriste na pravom ispitnom skupu kako bi se dobili konačni rezultati.

Drugi stupac u tablici 4.18 predstavlja najbolje rezultate koji su dobiveni pomoću LSTM-a ili GRU-a.

Tablica 4.18: Usporedba najboljih rezultata LSTM/GRU-a i SVM-a

	SVM	LSTM/GRU
Obama	57.8%	**61.5%**
Prava homoseksualaca	**64.9%**	59.5%
Marihuana	**73.4%**	69.9%
Pobačaj	**62.0%**	56.6%
Pauza.hr	**87.2%**	83.5%

SVM pokazuje bolje rezultate u četiri od pet skupova podataka, no treba uzeti u obzir da RNN-ovi i sve njegove podvrste traže veliku količinu podataka kako bi pokazali svoje najbolje performanse. Mreže su trenirane u prosjeku na oko tisuću primjera što je zanemarivo u odnosu na IMDb skup podataka u kojem je dostupno 50000 primjera. U radu studenata sa Stanforda [10] se pokazalo da je LSTM vrlo uspješan u klasifikaciji IMDb skupa podataka što sugerira da je potreban puno veći skup kako bi se postigli rezultati koji su bolji od rezultata klasičnih algoritama strojnog učenja.

Iako rezultati za LSTM i GRU ukupno gledajući nisu bolji od SVM-a, može se vidjeti da čak i sa malom količinom podataka, duboke mreže daju solidne rezultate koji su udaljeni u prosjeku za 4.5% u primjerima gdje je SVM bolji od LSTM-a i GRU-a.

[10] Korištena je ista podjela ispitnog skupa i skupa za učenje.

5. Zaključak

Teorija klasičnih neuronskih i dubokih neuronskih mreža je odavno utemeljena, no tek se nedavno počela iskorištavati njihova puna moć. Iako se još uvijek smišljaju nove vrste dubokih mreža koje se temelje na klasičnim mrežama, LSTM i GRU danas predstavljaju vrhunac dostignuća u području dubokih mreža. Pokazali su iznimne rezultate u raznim primjenama, a pogotovo u području strojnog prevođenja, odnosno općenitije u području prirodnog jezika.

Ovim radom istražena je primjena modela dubokog učenja u otkrivanju stavova u korisničkim komentarima na engleskom i hrvatskom jeziku. U eksperimentima su ispitane razne konfiguracije mreža te su nađeni optimalni parametri koji daju najbolje rezultate u svakom od skupova podataka. Ispitan je i utjecaj procesiranja podataka i empirijski se pokazalo da lematizacija nije dobar odabir kada su u pitanju mreže LSTM ili GRU. Ispostavilo se da je veličina vremenskog sloja vrlo bitan parametar koji ovisi o samoj strukturi podataka. Ako se koristi veličina koja je daleko veća od prosječne duljine teksta, mreža će se pretrenirati. U slučaju da je veličina premalena, mreža će se podtrenirati i neće dostići maksimalne moguće performanse.

Utjecaj ostalih parametara mreže je također bitan, no ne postoji univerzalno pravilo za sve vrste podataka. Svaki skup podataka je davao optimalne rezultate sa različitim parametrima. Razlika između ćelija GRU i LSTM je rezultirala u razlici preciznosti svega nekoliko postotaka i to u korist LSTM-a.

Iako se u eksperimentima SVM pokazao superiornim, mora se uzeti u obzir količina podataka koja je bila dostupna. Uz već naveden rad studenata sa Stanforda na IMDb skupu podataka, LSTM-ovi su pokazali i superiorne rezultate u klasifikaciji s više od dvije klase [22], no bitno je naglasiti da je bio dostupan puno veći uzorak podataka.

LITERATURA

[1] Dzmitry Bahdanau, Kyunghyun Cho, i Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Shamim Biswas, Ekamber Chadda, i Faiyaz Ahmad. Sentiment analysis with gated recurrent units.

[3] Erik Cambria i Amir Hussain. *Sentic computing: a common-sense-based framework for concept-level sentiment analysis*, svezak 1. Springer, 2015.

[4] Lipika Dey i SK Mirajul Haque. Opinion mining from noisy text data. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):205–226, 2009.

[5] John Duchi, Elad Hazan, i Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[6] Goran Glavaš, Damir Korencic, i Jan Šnajder. Aspect-oriented opinion mining from user reviews in croatian. *ACL 2013*, stranica 18, 2013.

[7] Jiang Guo. Backpropagation through time. 2013.

[8] Kazi Saidul Hasan i Vincent Ng. Stance classification of ideological debates: Data, models, features, and constraints.

[9] Sepp Hochreiter i Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] James Hong i Michael Fang. Sentiment analysis with deeply learned distributed representations of variable length texts.

[11] Diederik Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, i Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[13] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, i Christopher Potts. Learning word vectors for sentiment analysis. U *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, stranice 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

[14] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, i Sanjeev Khudanpur. Recurrent neural network based language model. U *INTERSPEECH*, svezak 2, stranica 3, 2010.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, i Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Razvan Pascanu, Tomas Mikolov, i Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[17] David E Rumelhart, James L McClelland, PDP Research Group, et al. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1-2. *Cambridge, MA*, 1986.

[18] David E Rumelhart, Geoffrey E Hinton, i Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[19] Haşim Sak, Andrew Senior, i Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, i Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Ryan A Stevenson, Joseph A Mikels, i Thomas W James. Characterization of the affective norms for english words by discrete emotional categories. *Behavior Research Methods*, 39(4):1020–1024, 2007.

[22] Duyu Tang, Bing Qin, i Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. U *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, stranice 1422–1432, 2015.

[23] T Tieleman i G Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. Technical report, Technical report, 2012.

Dodatak A
Upute za instalaciju i pokretanje

Postavljanje projekta se svodi na instalaciju potrebnih programa koji su korišteni tijekom izrade. Svi navedeni postupci za instalaciju se odnose na distribuciju Arch Linux.[1] Korišteni programi su:

- *Python3* — skriptni programski jezik koji omogućava brz razvoj složenih sustava
- *pip* — upravljač paketima za programski jezik *Python*
- *virtualenv* — alat za kreiranje virtualnih okruženja

A.1. Instalacija programskog jezika *Python* i pripadajućih komoponenti

Prije same instalacije je poželjno osvježiti sve repozitorije naredbom `#pacman -Syyu` kako bi se instalirale najnovije inačice programa. *Python3* se jednostavno instalira pokretanjem naredbe `#pacman -S python`

pip[2] je alat koji služi za instalaciju dodatnih paketa za programski jezik *Python*. Njegova najveća prednost je jednostavnost upravljanja paketima te u kombinaciji sa alatom *virtualenv* znatno ubrzava razvoj rješenja u *Pythonu*.

pip se instalira naredbom `#pacman -S python-pip`

virtualenv[3] je alat koji omogućava stvaranje izoliranih Pyhton okruženja. Primjerice, ako radite na više projekata na jednom računalu, a svaki od tih projekata zahtijeva različite verzije paketa za *Python*, onda svakom projektu dodijelite vlastito okruženje

[1] `https://www.archlinux.org/`
[2] `https://pypi.python.org/pypi/pip`
[3] `https://virtualenv.pypa.io/en/latest/`

u koje se instaliraju odgovarajući paketi.
Instalacija se vrši pokretanjem naredbe `#pacman -S python-virtualenv`

A.1.1. Instalacija potrebnih *pip* paketa

Prije same instalacije paketa, potrebno je stvoriti novo virtualno okruženje gdje će se instalirati svi potrebni paketi. Pozicionirajte se u proizvoljan direktorij na računalu, kao što je npr. `/home/ime_korisnika/virtualna_okruzenja/`. Nakon toga stvorite novo virtualno okruženje pomoću naredbe `virtualenv ime_okruzenja`.

Sada je okruženje spremno za uporabu i treba ga aktivirati kako bi *pip* znao da treba instalirati sve pakete u direktorij gdje se nalazi stvoreno virtualno okruženje: `source /putanja/do/virtualnog/okruzenja/bin/activate`.

Instalacija paketa vrši se naredbom
`pip -r install korijen_projekta/src/requirements.txt`.
Potrebne pakete je moguće instalirati jednom naredbom jer su svi paketi navedeni u datoteci `requirements.txt`, stoga ih *pip* može instalirati direktno iz te datoteke.

Izuzetak je paket *tensorflow* koji se mora instalirati naredbom[4]

```
pip install --upgrade https://storage.googleapis.com/
tensorflow/linux/gpu/tensorflow-0.8.0-cp34-cp34m-linux_x86_64.whl
```

Nakon instaliranja svih paketa, potrebno je pokrenuti sljedeću naredbu:
`python -m spacy.en.download`. Ovo će preuzeti datoteke koje su potrebne za engleski vokabular *spacy* paketa. Datoteke imaju oko 500MB.

NAPOMENA

Ove upute su namijenjene *Python* verziji 3.5, no projekt je razvijan na verziji 3.4. Razlog tome je paket *tensorflow*[5] koji trenutno (Ožujak 2016.) nema podršku za *Python* 3.5. Za verziju 3.4, upute se mijenjaju do te mjere da se treba instalirati *Python* 3.4 i odgovarajuća *pip* verzija. *virtualenv* alat ne ovisi o verziji *Pythona* i može se instalirati po gore navedenom postupku.

[4] Ova naredba vrijedi samo za trenutnu verziju paketa. U slučaju izdavanja nove verzije treba se otići na službenu stranicu i preuzeti novu poveznicu.

[5] `https://www.tensorflow.org/`

A.2. Pokretanje

Prije pokretenja učenja mreže je potrebno generirati serijalizirane podatke iz sirovih podataka. Ovaj pristup je odabran jer je za procesiranje sirovih podataka uvijek potrebno učitavati razne vokabulare koji uzimaju mnogo memorije i vremenski im dugo treba da se učitaju.

Izvršna datoteka se zove `main.py` te se pokreće naredbom:
`python main.py [dodatne_opcije] [argumenti]`

A.2.1. Dodatne opcije i argumenti

Dodatne opcije su vezane uz način procesiranja podataka, dok se argumentima bira domena na kojoj se želi provoditi učenje. Dodatne opcije su:

- *-l* (--load_model) — Određuje hoće li se koristiti predtrenirani model
- *-p* (--process_data) — Radi procesiranje sirovih podataka i sprema ih kao serijalizirane datoteke
- *-s* (--standard_ml) — Ako se upali ova opcija, koristi se treniranje pomoću SVM algoritma. Duboko učenje se ne provodi

Prilikom prvog pokretanja se mora upaliti *-p* opcija kako bi se generirali potrebni serijalizirani podaci. Opcija *-l* može se upaliti samo ako smo prije učenja nad domenama pokrenuli naredbu:
`python main.py -pl imdb_movies_train [veličina_batcha] [broj_epoha]`
Time će se generirati *tensorflow* model koji je treniran na IMDb skupu podataka. U slučaju da se postavi opcija *-s*, model će se samo trenirati sa SVM-om i zanemarit će ostale opcije.

Argumenti koji se mogu zadati su:

- *domena* — Na kojoj domeni će se model trenirati
- *veličina batcha*
- *broj epoha*

Primjer zadavanja naredbi je moguće vidjeti u skripti `start_script.sh`. Dostupna je i pomoćna opcija koja ispisuje način korištenja skripte: `python main.py -h`.

Primjena modela dubokog učenja za otkrivanje stavova u korisničkim komentarima

Sažetak

Duboko učenje je relativno nova grana strojnog učenja koja se temelji na teoriji neuronskih mreža. Pomoću naprednih arhitektura dubokih neuronskih mreža danas se pokušavaju riješiti poznati problemi u području obrade prirodnog jezika kao što su analiza stavova i sentimenta. U sklopu završnog rada napravljene su dvije poznate konfiguracije povratnih neuronskih mreža LSTM i GRU pomoću programa *Tensorflow*. Provedeni su razni eksperimenti u kojima se traže optimalni parametri povratnih neuronskih mreža s ciljem točne klasifikacije stavova u korisničkim komentarima iz skupova podataka na hrvatskom i engleskom jeziku.

Ključne riječi: umjetna neuronska mreža, duboko učenje, LSTM, GRU, strojno učenje, obrada prirodnog jezika

Application of deep learning for stance detection in user comments

Abstract

Deep learning is a relatively new branch of machine learning based on the theory of neural networks. Using the advanced deep neural network architectures, today's well-known natural language processing problems like sentiment and opinion mining are being solved. Two widely used configurations of recurrent neural networks were created with the *Tensorflow* software: LSTM and GRU. Many different experiments were conducted in which the focus was on finding the optimal parameters of the recurrent neural networks, which would yield in correct stance classification of user comments from the Croatian and English datasets.

Keywords: artificial neural network, deep learning, LSTM, GRU, machine learning, natural language processing