Web aplikacija za analizu sentimenta na društvenim mrežama

Košutić, Denis

Master's thesis / Specijalistički diplomski stručni

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:607097

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College

VISOKO UČILIŠTE ALGEBRA

DIPLOMSKI RAD

Web aplikacija za analizu sentimenta na društvenim mrežama

Denis Košutić

Zagreb, rujan 2019.

Predgovor

Zahvaljujem se svome mentoru v.pred. Aleksandru Radovanu, dipl. ing. koji mi je

pomogao oblikovati ideju i imao puno strpljenja i stručnih savjeta tijekom izrade diplomskog rada.

Također se zahvaljujem kolegama i kolegicama od kojih sam puno naučio kroz diplomski studij.

Posebno se zahvaljujem svojim roditeljima koji su me podupirali kroz cijelo školovanje i omogućili ovaj studij. Bez njih moja dostignuća ne bi bila moguća.

Hvala vam od srca!

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme diplomskog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Društvene mreže predstavljaju glavni medij za izražavanje mišljenja i stavova o raznim područjima – kvaliteti proizvoda, popularnosti političke stranke, imidžu organizacije i slično. Svakodnevno se generiraju velike količine podataka na društvenim mrežama, koje je moguće analizirati i izvući korisne informacije. Sentiment je ideja, mišljenje ili osjećaj o nekoj temi. Analiza sentimenta je automatizirani proces razumijevanja mišljenja i stavova iz pisanog ili govornog jezika koristeći različite koncepte i algoritme. Cilj takvog procesa je utvrđivanje jesu li mišljenja izražena u tekstu pozitivna, negativna ili neutralna. Tradicionalno se analiza teksta temeljila na većim tekstovima kao što su članci ili recenzije. U današnje doba društvene mreže, blogovi i web portali izlažu korisnicima mogućnost izražavanja mišljenja u kratkim komentarima. Prema tome se oblikuju i algoritmi kako bi preciznost klasifikacije sentimenta bila što točnija. Ovaj radi temelji se na usporedbi različitih algoritama i odabiru optimalnog rješenja za klasifikaciju sentimenta. Za treniranje algoritama korišten je IMDb skup podataka. Izneseni su rezultati i dobivene preciznosti pojedinog algoritma. Izrađena je Python aplikacija za klasifikaciju sentimenta, te web aplikacija koja preko korisničkog sučelja prikazuje statistiku pozitivno i negativno klasificiranih komentara s društvene mreže Twitter.

Ključne riječi: društvene mreže, analiza sentimenta, klasifikacija, Python, Twitter.

Social networks are the main medium for expressing opinions and views on various fields - product quality, political party popularity, brand reputation and others. Large amounts of data are generated daily on social networks, which can be analyzed and useful information can be extracted. Sentiment is an idea, opinion, or feeling about a topic. Sentiment analysis is an automated process of understanding opinions and thoughts within written or spoken language using different concepts and algorithms. The purpose of that process is to determine whether opinions expressed in the text are positive, negative or neutral. Traditionally, text analysis has been based on larger texts such as articles or reviews. Nowadays social networks, blogs and web portals expose users to express their opinions in short comments. Accordingly, algorithms are designed to make the classification of

sentiment more accurate. This paper is based on the comparison of different algorithms and the selection of the optimal solution for sentiment classification. The IMDb dataset was used to train the algorithms. The results and the precision of each algorithm are presented. A Python sentiment classification application was created, as well as a web application that displays statistics of positively and negatively classified Twitter comments via user interface.

Keywords: social networks, sentiment analysis, classification, Python, Twitter.

Sadržaj

1. Uvod 1

2. Analiza sentimenta 2

3. Algoritmi i modeli 3

3.1. Analiza teksta 3

3.2. Leksikon 4

3.3. Naivni Bayesov klasifikator 5

3.4. Algoritam maksimalne entropije 6

3.5. SVM 7

3.6. Neuronske mreže 8

3.6.1. Duboke neuronske mreže 9

3.6.2. LSTM 10

4. Implementacija modela i treniranje algoritama 12

4.1. Skup podataka 12

4.2. Implementacija 12

4.3. Priprema podataka 13

4.4. Leksikon 14

4.5. Naivni Bayesov klasifikator 16

4.6. Algoritam maksimalne entropije 17

4.7. SVM 18

4.8. Neuronske mreže 18

4.8.1. Duboke neuronske mreže 19

4.8.2. LSTM 21

5. Analiza sentimenta na hrvatskom jeziku 23

6. Web aplikacija 24
6.1. Django REST 25
6.2. Twitter API 27
6.3. Django Channels 29
6.4. React JS 30
6.5. Sučelje 32
Zaključak 35
Popis kratica 36
Popis slika 37
Popis tablica 38
Popis kôdova 39
Literatura 40
Prilog 43

1. Uvod

Svakodnevno se stvaraju velike količine nestrukturiranih podataka. Mogućnost analize i obrade podataka postaje sve važnija. Kompetitivna okolina u poslovanju zahtijeva brzo reagiranje na promjenjive zahtjeve kupaca i okolinske uvjete. Mišljenje kupaca o proizvodima i povratna informacija jedna je od najbitnijih faktora uspješnog poslovanja i reputacije organizacije. Obrada takvih podataka idealan je kandidat za primjenu strojnog učenja i automatizacije.

Ovaj rad bavi se tehnikama analize nestrukturiranog teksta i dobivanja korisnih informacija, posebice mišljenja i stavova ljudi o nekoj temi, odnosno analizu sentimenta. Takve tehnike spadaju pod područje obrade prirodnog jezika (engl. *natural language processing*, skraćeno NLP).

U današnje doba postoje mnoge platforme gdje korisnici mogu izraziti svoje mišljenje: web portali, blogovi, web stranice, društvene mreže. Jedna od najpopularnijih društvenih mreža na kojoj korisnici ostavljaju kratke komentare je Twitter. Na Twitteru poruke su ograničene na 280 znakova, što je idealno za analizu zbog velikog broja uzoraka. Moguće je izvući povratne informacije milijuna korisnika i dobiti njihov stav i osjećaj prema bilo kojem proizvodu ili usluzi i iskoristiti te podatke i informacije za buduće analize tržišta i poslovnih domena.

U ovom radu izrađena je Python aplikacija u kojoj su razvijeni algoritmi potrebni za analizu sentimenta nad podacima prikupljenih s društvene mreže Twitter. Isto tako je razvijena i web aplikacija koja predstavlja korisničko sučelje. Postoje poznati skupovi podataka na kojima se trenira model za analizu sentimenta i svake godine postižu se sve bolji rezultati. Za ovaj rad odabran je skup podataka IMDb recenzija filmova. Algoritmi pogodni za ovakvu analizu su: leksikon, naivni Bayesov klasifikator (engl. *naive Bayes classifier*), model potpornih vektora (engl. *support vector machine*, skraćeno *SVM*), algoritam maksimalne entropije (engl. *maximum entropy*) i neuronske mreže (engl. *neural networks*). Koristeći ove algoritme uspoređena je preciznost na navedenom skupu podataka. Prikazani su izazovi primjene dobivenog modela na hrvatski jezik i izneseni su rezultati testiranja aplikacije.

2. Analiza sentimenta

Sentiment je ideja, mišljenje ili osjećaj o nekoj temi. Analiza sentimenta (engl. *sentiment analysis*) je automatizirani proces razumijevanja mišljenja i stavova iz pisanog ili govornog jezika koristeći različite koncepte i algoritme. (Pang et al., 2008.)

Ljudi su u stanju podsvjesno klasificirati tekst kao pozitivan ili negativan. Primjerice, rečenica „Danas je unatoč lošem vremenu bio dobar dan“ većini će pružiti pozitivan stav, iako se ne sastoji samo od pozitivnih dijelova. Na takav zaključak dolazi se ispitivanjem riječi i uspoređivanjem pozitivnih i negativnih stavova. Riječ „dobar“ u kombinaciji s „je“ i „bio“ presudio je odluku o osjećaju, iako je fraza „lošem vremenu“ negativna komponenta. Taj proces događa se bez razmišljanja i možda se čini jednostavno, no u slučaju računalne obrade, ovo je veoma složen problem.

Primjena analize sentimenta provlači se u mnoga područja. Može se pratiti:

- Reakcija kupaca na proizvod
- Imidž organizacije
- Kvaliteta korisničke podrške
- Zadovoljstvo zaposlenika
- Analiza konkurencije
- Popularnost političke stranke ili osobe
- Otkrivanje nadolazeće krize

Možda najbolji primjer koji je pokrenuo upotrebu analize sentimenta u javnom sektoru je slučaj iz 2012. kada je Obamina administracija iskoristila ovu tehniku prije predsjedničkih izbora i dobila uvid u reputaciju Barracka Obame i Mitt Romneya, i pokazalo se da je Obama daleko u prednosti na socijalnim mrežama u odnosu na Romneya (Ronna, M., 2012). Nakon toga većina političkih kampanja koristi ovu tehniku kako bi pratila zadovoljstvo glasača i reagirala na vrijeme.

Danas gotovo svaka velika organizacija koristi sustav za analizu sentimenta kako bi dobila povratnu informaciju o proizvodu ili usluzi koju pruža. Tako Google ima *Natural Language* u sklopu Google Cloud-a, Amazon koristi *Amazon Comprehend* preko AWS-a (engl. *Amazon web services*, skraćeno AWS), IBM ima *Watson Natural Language Understanding* itd.

3. Algoritmi i modeli

Analiza sentimenta, kao i mnogi drugi NLP problemi, može se modelirati kao klasifikacijski problem u kojem se moraju riješiti dva zadatka:

- Razvrstavanje rečenica kao subjektivna ili objektivna, poznato kao klasifikacija subjektivnosti.
- Razvrstavanje rečenica kao izražavanje pozitivnog, negativnog ili neutralnog mišljenja, poznato kao klasifikacija polarnosti (engl. *polarity*)

Algoritmi se mogu svrstati u dvije skupine:

- Modeli temeljeni na brojanju pojava riječi
- Modeli temeljeni na nadziranom strojnom učenju

3.1. Analiza teksta

Obrada teksta počinje razdvajanjem dobivenog uzorka na manje dijelove kako bi se moglo analizirati pojedine riječi ili fraze.

Ova tehnika poznata je kao vreća riječi (engl. *bag of words*) i koristi se u većini algoritama u obradi prirodnog teksta. (Zhang et al., 2010.). Temelji se na odvajanju i određivanju frekvencije pojavljivanja svake pojedine riječi. Primjer rečenice:

„Ivan voli gledati filmove. I Petar voli gledati filmove“. Prikaz ove rečenice bio bi:

{ „Ivan“ : 1, „voli“:2, „gledati“: 2, „filmove“ :2, „i“ :1, „Petar“:1, }

Svakoj riječi pridijeljen je broj koji označuje koliko se ta riječ puta ponovila u zadanom tekstu. Poredak riječi nije očuvan. Ovaj način razdvajanja je pogodan za spajanje više uzoraka tako da se napravi unija između njih.

Na sličan način dobiva se razdioba po više od 1 riječi, što se označuje kao n-gram model.

U slučaju razdvajanja po dvije riječi, model bi se zvao bigram i na primjeru prethodne rečenice izgledao bi ovako:

{ „Ivan voli“ : 1, „voli gledati“:2, „gledati filmove“: 2, „I Petar“ :1, „Petar voli“:1, }

U sljedećim poglavljima opisane su tehnike analize sentimenta primijenjene u ovom radu.

3.2. Leksikon

Leksikon je baza riječi gdje svaka riječ ima pridijeljen sentiment – pozitivan ili negativan koeficijent. Takav rječnik se onda koristi kako bi se dobio sveukupni koeficijent sentimenta u zadanom tekstu. Formula za izračun glasi:

$$k = f(\sum pozitivni_koeficijenti \; - \; \sum negativni_koeficijenti) \qquad (1)$$

Prolazi se kroz zadani tekst, provjeravaju se sve riječi i ako postoje u leksikonu, zbraja se njihov koeficijent (pozitivan i negativan) te se dobiva konačan rezultat koji se tada provlači kroz funkciju normalizacije.

Primjer isječka iz CroSentiLex leksikona (Sentilex, 2012.):

Tablica 1. Leksikon

Negativno	koeficijent	Pozitivno	koeficijent
rat	1	lijep	1
okriviti	0.92307	neopisiv	0.7926
užasan	0.82494	kompliment	0.77335
jeziv	0.79414	olakšanje	0.75785
divljanje	0.76832	opuštanje	0.72478

Cataldo Musto (Musto et al., 2014.) u svome radu piše kako se analiza pomoću leksikona temelji na uvidu da se polarnost dijela teksta može dobiti na odnosu polariteta pojedine riječi koje ga čine. Međutim, zbog složenosti prirodnih jezika, tako jednostavan pristup vjerojatno neće uspjeti s obzirom na to da se mnogi aspekti ne uzimaju se u obzir (npr. prisutnost negacije, neispravni oblici riječi, sarkazam itd.) Kao posljedicu toga, predlaže precizniji pristup: razlaganje zadanog teksta na mikro fraze, koje bi se dijelile prema interpunkcijskim znakovima, te računanju koeficijenta sentimenta odvojeno po svakoj mikro frazi. Svaki put kada se u mikro frazi nađe negacija, koeficijent sentimenta se okreće u drugom smjeru. Tako se dobivaju bolji rezultati od jednostavnog prebrojavanja frekvencije riječi.

3.3. Naivni Bayesov klasifikator

Naivni Bayes jednostavna je tehnika za konstrukciju klasifikatora; modela predstavljenih kao vektori vrijednosti značajki, gdje su oznake klase izvedene iz nekog konačnog skupa. Ne postoji niti jedan algoritam za kreiranje takvih klasifikatora, već postoji obitelj algoritama utemeljena na zajedničkom principu: svi naivni Bayesovi klasifikatori pretpostavljaju da vrijednost određenog svojstva nije ovisna o vrijednosti bilo kojeg drugog svojstva, s obzirom na ciljanu varijablu. (Dinu et al., 2012.) Svi klasifikatori se temelje na Bayesovom teoremu:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

Koristeći teorem može se izračunati vjerojatnost događaja A uz uvjet da se dogodio događaj B. U slučaju analize teksta, provjerava se vjerojatnost pojavljivanja riječi unutar zadanog teksta. Kreiranje klasifikatora za analizu sentimenta odvija se u sljedećim koracima:

- Set za treniranje klasifikatora sastoji se od uzoraka za koje se zna ishod – negativan ili pozitivan sentiment
- Svaki uzorak se rastavlja na vreću riječi
- Pretprocesiranje – čišćenje uzoraka (neispravne riječi)
- Svaka riječ ili fraza ima vjerojatnost pojave u uzorku
- Postoji naivna pretpostavka da je svaka riječ ili fraza nezavisna u odnosu na druge riječi ili fraze
- Svaka vreća riječi ima pridijeljenu određenu vjerojatnost da daje negativan ili pozitivan rezultat
- Tako posloženi podaci koriste se u postupku treniranja modela

Model računa sentiment uzorka na temelju vjerojatnosti pojave određene kombinacije riječi koje su istrenirane unaprijed. Svaka riječ se zbraja kao negativna ili pozitivna vjerojatnost i tako se dobiva rezultat.

Metoda se naziva naivnom jer pretpostavlja nezavisnost između značajki, odnosno riječi koje utječu na rezultat. Iako u pravim podacima to nije slučaj i riječi ovise jedna o drugoj, ova metoda i dalje daje dobre rezultate. Prednost naivnog Bayesovog klasifikatora je u tome što zahtjeva vrlo mali set za treniranje u odnosu na druge metode. (Go et al., 2009.)

3.4. Algoritam maksimalne entropije

Entropija je prosječna količina informacija koja se nalazi u svakoj poruci (Rao et al., 2015.). Raspodjela vjerojatnosti događaja, te količina informacija svakog događaja, tvori slučajnu varijablu čija je očekivana vrijednost prosječna količina informacija generirana ovom distribucijom. Entropija događaja koji je siguran da će se dogoditi jednaka je nuli jer ne donosi nikakvu korisnu informaciju.

Cilj je iskoristiti kontekstualne informacije u uzorku (riječ ili fraze unutar teksta) kako bi odredili koeficijent sentimenta (Pang, 2002.). Kreiranje modela se sastoji od sljedećih koraka:

- Sakupljanje podataka u formatu (x_i, y) gdje je x_i riječ ili fraza, a y je klasa
- Izračunati vjerojatnost distribucije

  $p(x_i, y) = \frac{1}{N}\ x\ broj\ puta\ koliko\ se\ (x_i, y)\ pojavi\ u\ uzorku$, N- broj uzoraka

- Izračunata vjerojatnost služi za pridruživanje klasa uzorcima ovisno o kontekstualnim informacijama
- Određivanje normalizacijske funkcije

  $$f(x_i, y) = \begin{cases} 1\ ako\ je\ y = klasa\ i\ uzorak\ sadrži\ riječ \\ 0, inače \end{cases}$$

- Treniranje modela

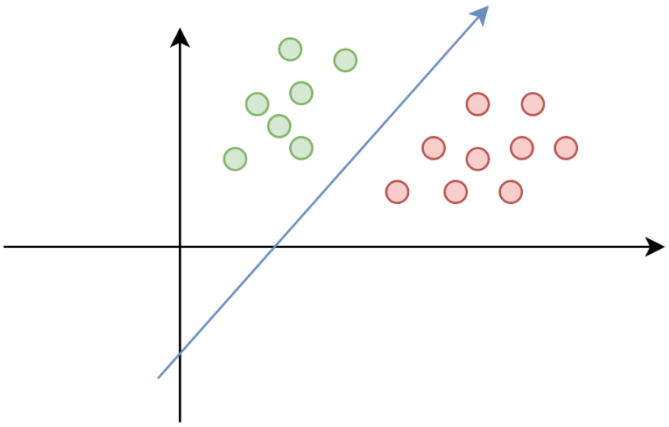
Ako se svaki uzorak (x_i, y) pojavi samo jednom, onda je $p(x_i, y)$ jednaka $\frac{1}{N}$.

Za razliku od naivnog Bayesovog klasifikatora, algoritam maksimalne entropije ne donosi pretpostavku o vjerojatnosti pojave pojedine riječi u uzorku. Glavni princip algoritma maksimalne entropije je da vjerojatnost distribucije treba biti uniformna u slučaju kada vjerojatnost nije poznata unaprijed.

Algoritam maksimalne entropije se koristi u mnogim problemima obrade prirodnog jezika i u nekim slučajevima daje bolje rezultate od naivnog Bayesovog klasifikatora kad je u pitanju analiza teksta. (Pang, 2002.)

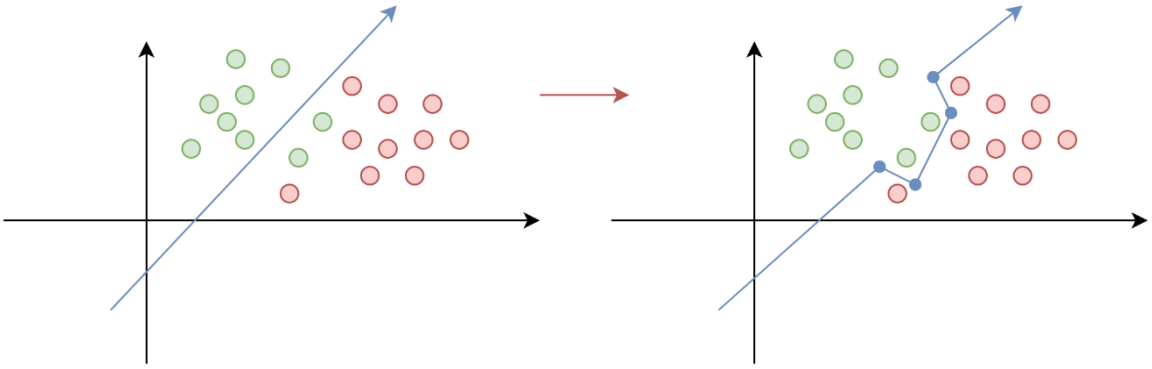
3.5. SVM

Model potpornih vektora (engl. *Support vector machines,* skraćeno SVM) je klasifikacijski model koji pomoću n-dimenzionalne plohe (engl. *hyperplane*) razdvaja set podataka na klase. (Medium, 2017.)



Slika 1. SVM prikaz razdvajanja skupa podataka

Na Slika 1. može se vidjeti vektor koji razdvaja skup podataka na dva dijela, odnosno na dvije klase. Moguće je odrediti više vektora koji bi podijelili prikazani skup podataka, a cilj SVM-a je odrediti onaj koji ima maksimalnu marginu, odnosno maksimalnu udaljenost između točaka jednog i drugog skupa. Problem nastaje kada se skupovi podataka preklapaju ili su u takvom položaju da ne postoji pravac koji bi razdvojio skup kao što se može vidjeti na Slika 2. s lijeve strane:



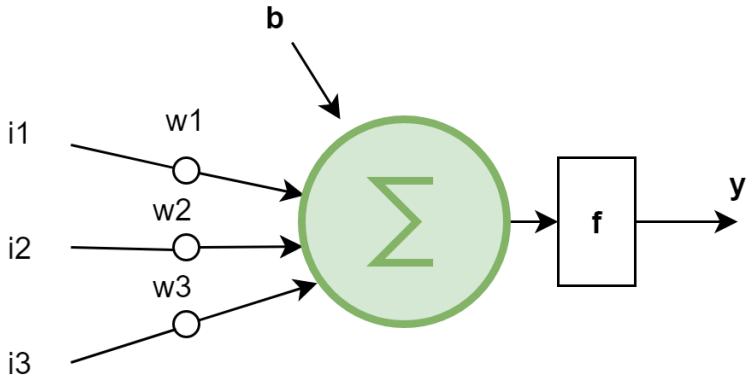
Slika 2. SVM prikaz regularizacije

Kako bi se riješio ovaj problem, uvodi se parametar regularizacije c (engl. regularization).

Što je taj parametar veći, to će model biti kompleksniji i obuhvatit će više podataka, no postoji opasnost kreiranja modela koji je specifičan za testne podatke (desna strana Slika 2).

3.6. Neuronske mreže

Neuronske mreže temelje se na modelu neurona koji su posloženi u više slojeva i vrlo su efikasni u strojnom učenju. (Bašić et al., 2008)



Slika 3. Umjetni neuron

Umjetni neuron se sastoji od ulaznih signala i_1, i_2...i_n koji su pomnoženi s težinama w_1,w_2,...w_n te zbrojeni s pragom b. Utezi utječu na aktivacijsku funkciju f i tijekom treniranja mreže to su parametri koji se mijenjaju u svakoj iteraciji. Zbrojeni ulazni signali prolaze kroz aktivacijsku funkciju i daju određeni izlaz. Izlazna funkcija ima oblik:

$$y = f(\sum_{k=1}^{n}(i_k w_k) + b) \tag{3}$$

Postoji više aktivacijskih funkcija, od kojih su najviše korištene:

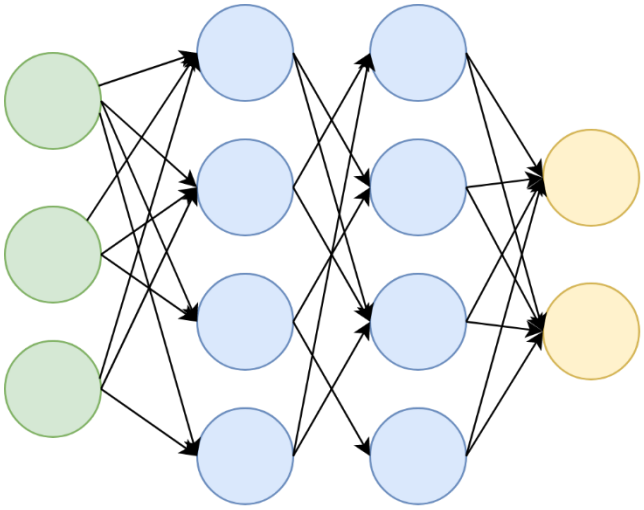
- Tangens-hiperbolna funkcija
- Logistička funkcija
- ReLu (engl. *rectified linear unit*)
- Softmax

Mreža se trenira kroz epohe i iteracije promjenom težina i pragova. Tako se dobiva set istreniranih parametara koji se koriste za predikciju sljedećeg uzorka. Proces treniranja odvija se kroz algoritam koji se naziva prolazak unatrag (engl. *backpropagation*). Pogreška mreže definirana je kao funkcija, pa se deriviranjem funkcije pogreške (engl. *cost function*) popravlja preciznost mreže. Brzina učenja ovisi o stopi učenja koja se može definirati i obično iznosi 0.001. Ako je stopa učenja prevelika, model neće postići optimalnu postavku težina, a ako je premala, imat će slabi utjecaj na izlaznu vrijednost.

3.6.1. Duboke neuronske mreže

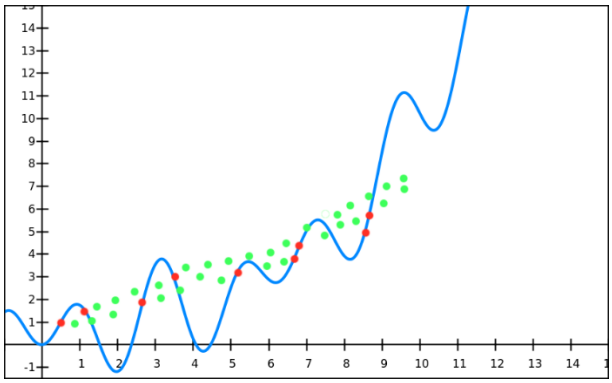
Duboke neuronska mreža sastoji se od:

- Ulaznog sloja
- Proizvoljnog broja skrivenih slojeva
- Izlaznog sloja



Slika 4. Duboka neuronska mreža

Slika 4. pokazuje duboku neuronsku mrežu sa jednim ulaznim, dva skrivena i jednim izlaznim slojem. Svaki čvor u mreži odgovara neuronu koji se ponaša kao n-dimenzionalna linearna regresija i prosljeđuje rezultat na sljedeći neuron u mreži. Skrivenih slojeva može biti proizvoljan broj, što određuje razinu detalja koju mreža razlikuje u zadanom skupu podataka. Jedan od problema neuronskih mreža je pretreniranje (engl. *overfitting* - Freitas, 2014.), što znači da će model biti previše specifičan za testne podatke i neće dobro raditi na novim uzorcima.



Slika 5. Pretreniranje

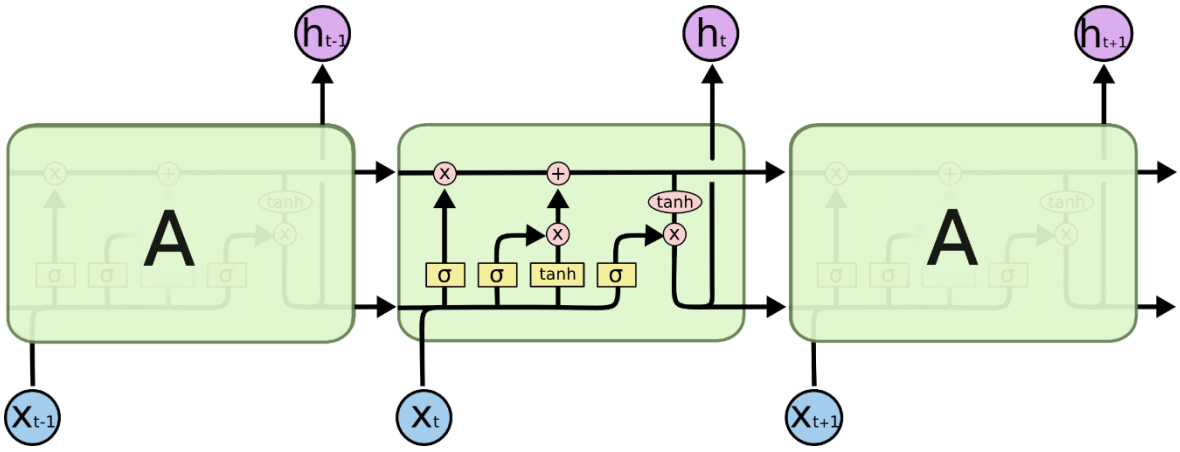
Na Slika 5. prikazana je funkcija $f(x) = \cos(2x) - 1.5\cos(3) + e^{0.3x}$ koja savršeno prolazi crvenim točkama prikazanim u koordinatnom sustavu. Nakon prikupljanja više podataka označenih zelenim točkama, može se zaključiti da navedena funkcija nije dobar model za ulazne podatke. Došlo je do pretreniranja, odnosno odabira prekompleksnog modela.

Postoje metode koje sprječavaju pretreniranje:

- Rano zaustavljanje (engl. *early stopping*)
- Unakrsna validacija (engl. *cross validation*)
- Izbacivanje neurona (engl. *dropout*)
- Umjetno povećavanje skup podataka (engl. *data augmentation*)

3.6.2. LSTM

Neuronske mreže s kratkotrajnim pamćenjem (engl. *Long-Short Term Memory*, skraćeno LSTM) su mreže koje spadaju pod rekurentne neuronske mreže (engl. *Recurrent neural network*, skraćeno RNN). Za razliku od dubokih neuronskih mreža, ovakve mreže mogu pamtiti informacije između iteracija koristeći module za ponavljanje, odnosno petlje.



[1]

Slika 6. LSTM neuronska mreža

Na Slika 6. prikazan je modul za ponavljanje u LSTM mreži. Crne linije prikazuju vektore koji spajaju čvorove u mreži. Rozi krugovi označuju operacije zbrajanja i multiplikacije nad vektorima. Žuti pravokutnici su slojevi u neuronskoj mreži. LSTM mreža ima mogućnost

[1] Slika 6 preuzeta je s web stranice (http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png)

dodavanja ili oduzimanja informacija svakom neuronu, što je regulirano preko vrata (engl. *gates*).

Prvi korak je određivanje koji dio informacije će se odbaciti iz stanja u neuronu. Ovo se odvija u sigmoidnom sloju koji se naziva sloj vrata zaboravljanja (engl. *forget gate layer*). Uzima se prethodna izlazna vrijednost h_{t-1} i trenutna ulazna vrijednost x_t, te se računa nova izlazna vrijednost koja je 0 ili 1. Vrijednost 0 označava zaboravljanje vrijednosti, a 1 označava da se vrijednost pamti.

Sljedeći korak je određivanje koji dio informacije će se dodati u stanje neurona. Ovo se odvija u sigmoidnom sloju koji se naziva sloj ulaznih vrata (engl. *input gate layer*). Kreira se vektor novih mogućih vrijednosti C_t koji se može dodati u trenutno stanje neurona.

U zadnjem koraku spajaju se prethodno stanje C_{t-1} i trenutno stanje C_t i odabire se što će biti izlazna vrijednost. Dobiveni vektor se provlači kroz funkciju tanh kako bi se normalizirale vrijednosti.

Na ulaz neurona ne dolazi samo trenutni ulaz, već i prethodno stanje neurona. Izlaz h_t može se izračunati sljedećim formulama (Wang et al, 2016.):

$$i_t = \sigma(W_o[h_t, x_t] + b_o) \tag{4}$$

$$f_t = \sigma\big(W_f[h_{t-1}, x_t] + b_f\big) \tag{5}$$

$$C_t = f_t * C_{t-1} + i_t * \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{6}$$

$$h_t = \sigma_t * \tanh(C_t) \tag{7}$$

Vrata *i* u trenutku *t* ovise o ulaznim podacima *x* i dodatnim parametrom *b*. Vrata *f* regulira količinu informacija koju funkcija propagira iz trenutka *t-1* i služi kao faktor zaboravljanja. Novo stanje dobiveno formulom h_t zapravo je superpozicija kombinacije prethodnog stanja, određenog faktora zaboravljanja i ulaznih podataka.

4. Implementacija modela i treniranje algoritama

4.1. Skup podataka

Kao skup podataka za treniranje algoritama odabran je IMDb skup recenzija filmova. Sastoji se od 25,000 recenzija za treniranje i 25,000 recenzija za testiranje. Sastavili su ga Andrew L. Maas i Raymond E. Daly 2011. godine koristeći podatke sa stranice *www.imdb.com (Maas et al*, 2011.*)*. Sadrži jednak broj pozitivnih i negativnih uzoraka. Ovaj skup omogućuje analizu kompleksnih fraza i pogodan je za analizu sentimenta. Uzorci su u obliku kratkog teksta što je vrlo slično komentarima na Twitteru. Skup podataka je podijeljen u 2 dijela:

- /train – podaci treniranje
- /test – podaci za testiranje

Primjer podataka se vidi u sljedećoj tablici:

Tablica 2. IMDb skup podataka

I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy.	positive
So im not a big fan of Boll's work but then again not many are.	negative

4.2. Implementacija

Za implementaciju i treniranje modela koristi se programski jezik Python (verzija 3.6.8).

Korištene Python biblioteke:

- TensorFlow – Glavna biblioteka za strojno učenje. Temelji se na protoku podataka kroz graf izračuna (engl. *computational graph*). Fleksibilna arhitektura biblioteke omogućava korištenje jednog ili više procesora ili grafičkih kartica
- Numpy – biblioteka za rad s matricama, odnosno N- dimenzionalnim nizovima.
- Scipy – biblioteka koja sadrži niz alata za rad s podacima i znanstvene izračune.
- Sklearn – biblioteka za strojno učenje koja sadrži mnogo modula za optimizaciju, statistiku i linearnu algebru

- Keras – biblioteka za strojno učenje i izradu prototipova
- Nltk – biblioteka za obradu prirodnog teksta

4.3. Priprema podataka

Priprema podataka jedan je od najbitnijih koraka u strojnom učenju (engl. *preprocessing*) što dovodi do boljeg modela i veće preciznosti. Podaci iz IMDb skupa podataka skupljeni su s web stranica što znači da se u uzorcima pojavljuju neispravni znakovi, dijelovi HTML-a i slično. U razvoju algoritama koriste se pomoćne funkcije koje čiste podatke u sljedećim koracima:

- čišćenje HTML tagova
- provjera interpunkcijskih znakova
- micanje zaustavnih riječi (engl. *stop words)* – riječi koje ne utječu na sentiment
- pretvorba u mala slova (engl. *lowercase*)

```
from bs4 import BeautifulSoup
from re import sub as replace
from nltk.corpus import stopwords
…
stopList = set(stopwords.words('english'))
def clean_data(text):
    no_html = BeautifulSoup(text).get_text()
    cleaned_interpunc = replace (r'[?|!|\'|"|#]',r'',
no_html)
    filtered_words = filter_stop_words(cleaned_interpunc)
    return filtered_words

def filter_stop_words(words):
    filtered_words =[]
    for word in words.split():
        if word not in stopList:
            filtered_words.append(word.lower())
    return filtered_words
```

Kôd 1. Priprema podataka

Kôd 1. prikazuje pripremu i čišćenje podataka. Naredbom `import` učitavaju se pomoćni moduli, funkcije i klase iz potrebnih biblioteka. Biblioteka `bs4` omogućuje parsiranje HTML dokumenata. HTML dokument ili tekst predaje se u konstruktor klase `BeautifulSoup` koji

koristeći parser pretvara tekst u stablo Python objekata. Nakon toga poziva se funkcija `get_text()` koja vraća samo tekstualni dio dokumenta bez HTML znakova.

Za micanje interpunkcijskih znakova koristi se regex koristeći funkciju `sub` iz modula `re`.

U analizi sentimenta postoje riječi koje su nebitne i nema utjecaj pa se koristi tehnika micanja takvih riječi prije analize. Za to se koristi lista riječi `stopwords` iz modula `nltk.corpus`. Isječak ove liste izgleda ovako:

```
{'out', 'with', 'they', 'own', 'an', 'be', 'some', 'for', 'do', 'its',
'yours', 'such', 'into', 'of', 'most', 'itself', 'other', 'off', 'is', 's',
'am', 'or', 'who', 'as', 'from', 'him', 'each', 'the', 'until', 'below',
'are', 'we', 'these', 'your', 'his', 'through', 'don', 'nor', 'me',
'this', 'our', ...}
```

Definira se funkcija `filter_stop_words(words)` koja u petlji prolazi kroz sve riječi u tekstu i odbacuje one koje se nalaze u navedenoj listi zaustavnih riječi. Prilikom dodavanja riječi koje odlaze na analizu koristi se funkcija `lower()` koja pretvara velika slova u mala.

Funkcijom `clean_data(text)` kombiniraju se sve navedene pretvorbe i tako se pripremaju ulazni podaci.

4.4. Leksikon

Odabran je leksikon SentiWords koji se sastoji od 155,00 engleskih riječi označenih s koeficijentom sentimenta između -1 i 1. (Gatti et al, 2016.). Dostupan je unutar ntlk Python biblioteke. Glavni isječak implementacije:

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn
from nltk.corpus import sentiwordnet as swn
from nltk import sent_tokenize, word_tokenize, pos_tag
…
lemmatizer = WordNetLemmatizer()
def calculate_sentiment(text):
   sentiment = 0.0
   sentences = sent_tokenize(text)
   for sentence in sentences:
       split_line = pos_tag(word_tokenize(sentence))
       for word, tag in split_line:
           wn_tag = penn_to_wn(tag)
```

```
                lemma = lemmatizer.lemmatize(word, pos=wn_tag)
                if not lemma:
                    continue
                synsets = wn.synsets(lemma, pos=wn_tag)
                if not synsets:
                    continue
                synset = synsets[0]
                swn_synset = swn.senti_synset(synset.name())
                sentiment += swn_synset.pos_score() -
swn_synset.neg_score()
   if sentiment >= 0:
       return 1
   return 0
```

Kôd 2. Leksikon

Kôd 2. prikazuje isječak analize sentimenta koristeći leksikon. Ulazni tekst dijeli se po rečenicama koristeći funkciju *sent_tokenize.* Nakon toga prolazi se po svim rečenicama i svaka rečenica se dijeli po riječima koristeći funkciju *word_tokenize.*

U tom trenutku ulazna rečenica „I thought this was a wonderful way to spend time on a summer holiday“ izgledala bi ovako: ['I', 'thought', 'this', 'was', 'a', 'wonderful', 'way', 'to', 'spend', 'time', 'on', 'a', 'summer', 'holiday'].

Iz modula `nltk.stem` uključuje se klasa `WordNetLemmatizer`. Ova klasa ima definiranu funkciju `lemmatize` koja analizira ulaznu riječ i pokušava pronaći njezin osnovni oblik. Ako ne uspije, vraća ulaznu riječ. Riječi se provjeravaju iz baze riječi `wordnet` koja dolazi iz modula `nltk.corpus`. U petlji se prolazi se po svakoj riječi i poziva se funkcija `lemmatize`. Ovo bi značilo da ako je ulazna riječ u množini, povratna vrijednost će biti u jednini. Nakon što je pronađen osnovni oblik, riječ se traži u SentiWords leksikonu funkcijom `synsets`. Ako je riječ pronađena, uzima se njezin koeficijent sentimenta definiran u leksikonu sentimenta i vraća se 1 ako je sentiment pozitivan, ili 0 ako je negativan.

Računa se preciznost na testnom skupu funkcijom `accuracy_score`:

```
from sklearn.metrics import accuracy_score
…
pred_y = [calculate(text) for text in test_X]
print accuracy_score(test_y, pred_y)
```

Koristeći leksikon dobiva se preciznost od 0.6917 , odnosno 69.17% na IMDb skupu. Ovaj rezultat je poprilično loš, ako se uzme u obzir da je preciznost od 50% nasumično

pogađanje. Ovo se može objasniti činjenicom da u recenzijama filmova postoji puno negacija i kombinacija negativnih i pozitivnih riječi, što nakon prebrojavanja često daje krive rezultate.

4.5. Naivni Bayesov klasifikator

Glavni isječak implementacije:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
…
def calculate_word_count(train_data, test_data):
    vectorizer = CountVectorizer(stop_words='english',
ngram_range=(1,3), max_features = 100000)
    train_features = vectorizer.fit_transform([r[0] for r in
train_data])
    test_features = vectorizer.transform([r[0] for r in
test_data])
   return train_features, test_features
…
def train_classifier(data, train_features):
   nb = MultinomialNB()
   nb.fit(train_features, [int(r[1]) for r in data])
   return nb
nb = train_classifier(data, train_features)
```

Kôd 3. Naivni Bayesov klasifikator

Kôd 3. prikazuje analizu sentimenta pomoću naivnog Bayesovog klasifikatora. Koristeći `CountVectorizer` klasu iz `sklearn.feature_extraction.text` modula kreira se mapa riječi i fraza iz ulaznog teksta s pripadajućim frekvencijama pojave. Parametar `ngram_range` označava raspon riječi u generiranim frazama prilikom treniranja modela, a parametar `max_features` određuje maksimalni broj kombinacija. Pozivom funkcije `vectorizer.fit_transform` učitavaju se podaci u format prikladan za treniranje modela.

Klasa `MultinomialNB` ima implementiran model naivnog Bayesovog klasifikatora i pozivom funkcije `fit` se trenira model. Predikcije sentimenta na testnim podacima računaju se s funkcijom `predict`. Nakon toga provjerava se preciznost modela:

```
from sklearn.metrics import accuracy_score
…
```

```
predictions = nb.predict(test_x)
print accuracy_score(test_y, predictions)
```

Preciznost modela na IMDb skupu je 0.8111, odnosno 81.11 %.

4.6. Algoritam maksimalne entropije

Glavni isječak implementacije:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
…
def train_classifier(data_train, labels):
   maxent = LogisticRegression(solver='sag',\
                               max_iter=1000,\
                               fit_intercept=True)
   maxent=maxent.fit(data_train, labels)
   return maxent
…
maxent = train_classifier(data_train, labels)
…
predictions = maxent.predict(test_x)
print accuracy_score(test_y, predictions)
```

Kôd 4. Algoritam maksimalne entropije

Kôd 4. prikazuje analizu sentimenta pomoću algoritma maksimalne entropije. Koristeći klasu `LogisticRegression` iz `sklearn.linear_model` modula kreira se model maksimalne entropije, koja je generalizacija modela logističke regresije (Mount, J., 2011.). Parametar `fit_intercept` označava da će se modelu dodati konstanta koja tijekom treniranja može utjecati na preciznost modela (engl. *bias*), `max_iter` označava broj iteracija, a `solver` označava algoritam optimizacije koji se koristi, što je u ovom slučaju stohastički gradijenti spust ili sag (engl. *Stochastic Average Gradient).* Model označen varijablom `maxent` trenira se pozivom funkcije `fit`. Preciznost testnog skupa testira se funkcijom `predict`. Dobivena preciznost na IMDb skupu je 0.8326, odnosno 83,26 %.

4.7. SVM

Glavni isječak implementacije:

```
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score
…
def train_classifier(data_train, labels):
   lsvm = LinearSVC()
   lsvm.fit(data_train, labels)
   return lsvm
…
lsvm = train_classifier(data_train, labels)
predictions = lsvm.predict(test_x)
print accuracy_score(test_y, predictions)
```

Kôd 5. SVM

Koristeći klasu `LinearSVC` iz `sklearn.svm` modula kreira se SVM model kojem se predaju ulazni podaci za testiranje. Treniranje modela odvija se pozivom funkcije `fit`. Predikcija na testnom skupu testira se pozivom funkcije `predict`.

Dobivena preciznost ovog modela je 0.7622 , odnosno 76,22%. Zaključuje se da je ovaj model prejednostavan za složenost zadanog skupa podataka.

4.8. Neuronske mreže

Neuronske mreže se za razliku od prethodnih jednostavnijih modela moraju implementirati u više koraka. Slaganje modela neuronskih mreža provedeno je koristeći Tensorflow biblioteku. Temelji se na protoku podataka kroz graf izračuna (engl. *computational graph*).

Kreiranje neuronske mreže odvija se kroz sljedeće korake:

- Dohvaćanje ulaznih podataka i pripadajućih oznaka (labela).
- Inicijalizacija algoritma/mreže:
    - Definiranje slojeva
    - Odabir funkcije pogreške
    - Odabir optimizacijske funkcije
- Inicijalizacija varijabli u grafu
- Pokretanje učenja mreže unutar petlje s određenim brojem iteracija.

4.8.1. Duboke neuronske mreže

Koristeći Keras biblioteku učitava se IMDb skup podataka:

```
from keras.datasets import imdb
top_words = 5000
(X_train, y_train), (X_test, y_test) =
imdb.load_data(num_words=top_words)
```

Skup podataka se učitava funkcijom `load_data` u kojoj se definira parametar `num_words=5000` koji određuje veličinu rječnika (engl. *vocabulary*), što znači da se iz uzoraka uzimaju samo prvih 5000 najčešće pojavljivanih riječi. Ovo je tehnika za pripremu podataka koja pomaže ukloniti riječi koje se pojavljuju rijetko, što bi samo unijelo šum u učenje mreže. Skupa podataka učitava se preko modula `keras.datasets`.

Neuronska mreža zahtjeva ulaze iste dimenzije. Provjerom prosječne veličine uzorka (recenzije filmova) dobiva se prosječni broj riječi 234,76. Pošto je razdioba eksponencijalna, uzima se 500 kao maksimalni broj riječi u sekvenci, što bi trebalo pokriti većinu skupa. Ako uzorak ima manje od 500 riječi, bit će nadomješten s 0 na kraju sekvence, a ako sadrži preko 500 riječi, bit će odrezan. To se postiže funkcijom `pad_sequences`:

```
from keras.preprocessing import sequence
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_words)
X_test = sequence.pad_sequences(X_test, maxlen=max_words)
```

Svaka riječ je reprezentirana kao 32-dimenzijski vektor, što se može vidjeti na Slika 7.`Slika 6`

```
[   4 3231  152  339    2   42 4869    2    2  345 4804    2  142   43
  218  208   54   29  853  659   46    4  882  183   80  115   30    4
  ...
  783    2   33    4 2945  103  465    2   42  845   45  446   11 1895
   19  184   76   32    4    2  207  110   13  197    4    2   16  601
   28    6   22   15  122   24 4171   72   33   32]
```

Slika 7. Vektorski prikaz riječi

Slaganje neuronske mreže započinje kreiranjem sekvencijalnog modela koristeći klasu `Sequential` iz modula `keras.models`. Slojevi se dodaju pozivom funkcije `add` iz navedene klase. Kao prvi sloj u mreži, koristi se sloj umetanja (engl. *Embedding layer*). Ovaj sloj služi za enkodiranje riječi u visoko dimenzionalne vektore, gdje se sličnost riječi translatira u blizinu između vektora u vektorskom prostoru. Ovi vektori se mijenjaju tijekom

učenja mreže i traži se optimalno mapiranje riječi i vektorske reprezentacije. Izlaz ovog sloja je vektor 32x500 dimenzija. Kôd 6. prikazuje slaganje slojeva neuronske mreže.

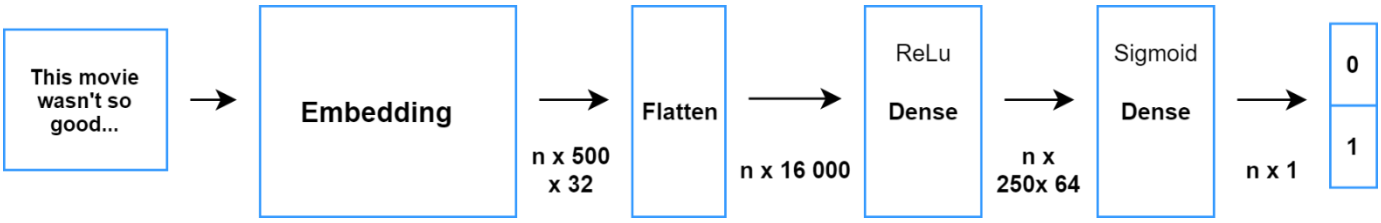
```
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers.embeddings import Embedding
…
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_words))
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

Kôd 6. Duboka Neuronska mreža

Nakon *Embedding* sloja, dodaje se sloj sažimanja (engl. *flatten*) koji će pretvoriti izlaz u jednu dimenziju. Nakon toga slaže se skriveni sloj s 250-dimenzionalnim izlazom. Kao aktivacijska funkcija koristi se ReLu (engl. *rectified linear unit*, skraćeno ReLu). Zadnji sloj daje 1-dimenzionalni izlaz, sa sigmoidnom aktivacijskom funkcijom koja će dati vrijednost između 0 i 1. Kao funkciju pogreške (engl. *loss function*) koristi se funkcija unakrsne entropije (engl. *cross entropy*), a optimizacijska funkcija je Adamov optimizator (napredna verzija gradijentnog spusta). Funkcija pogreške definira se pozivom funkcije `compile`:

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

Arhitektura mreže prikazana je na Slika 8.



Slika 8. Arhitektura duboke neuronske mreže

Model se trenira pozivom funkcije `fit`:

```
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=128)
```

Parametar *epochs* određuje u koliko epoha će se trenirati mreža, a `batch_size` određuje veličinu uzorka koji ulazi u svakoj iteraciji. Broj epoha ne smije biti prevelik kako ne bi došlo do pretreniranja (engl. *overfitting*).

Nakon treniranja ispituje se preciznost na skupu za testiranje funkcijom `evaluate`:

```
scores = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Dobivena je preciznost od 0.8988, odnosno 89,88 % .

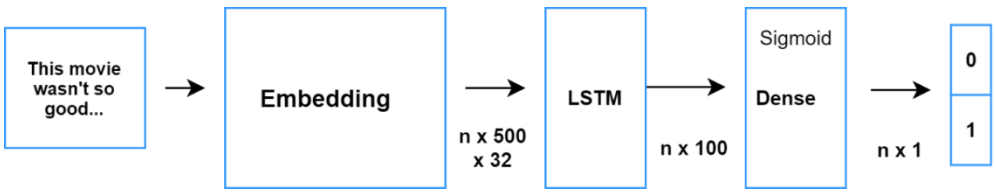
4.8.2. LSTM

LSTM mreža se razlikuje od duboke neuronske mreže po dodatnom LSTM sloju koji omogućuje pamćenje značajki tijekom procesa treniranja. Ostatak koda je sličan implementaciji duboke neuronske mreže i prikazan je u nastavku (Kôd 7):

```
from keras.models import Sequential
from keras.layers import Dense, Flatten, LSTM
from keras.layers.embeddings import Embedding
…
lstm_out = 100
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_words))
model.add(LSTM(lstm_out))
model.add(Dense(1, activation='sigmoid'))
…
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=10, batch_size=128)
…
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Kôd 7. LSTM neuronska mreža

Iz modula `keras.models` učitava se sekvencijalni model preko klase `Sequential`. Definira se parametar `lstm_out` koji određuje broj čvorova u LSTM sloju. Prvi sloj kao i kod duboke neuronske mreže je *Embedding* čiji je izlaz vektor dimenzija 500x32. Nakon toga slijedi LSTM sloj sa 100 memorijskih čvorova. Zadnji sloj je jednak kao i kod duboke neuronske mreže i također se koristi sigmoidna aktivacijska funkcija. Preciznost mreže na IMDb skupu je 0.9344, odnosno 93,44%. Arhitektura mreže prikazana je na Slika 9.



Slika 9. LSTM neuronska mreža

U tablici je prikaz usporedbe testiranih algoritama na IMDb skupu podataka:

Tablica 3. Usporedba algoritama

Algoritam	Preciznost %
Leksikon	69,17
Naivni Bayesov klasifikator	81,11
Algoritam maksimalne entropije	83,26
SVM	76,22
Duboka neuronska mreža	89,88
LSTM neuronska mreža	93,44

LSTM neuronska mreža postigla je najbolje rezultate od testiranih klasifikatora i ona će se koristiti u web aplikaciji. Tijekom učenja neuronske mreže moguće je spremiti stanje kako bi se kasnije koristio model (naučena mreža) i testirala klasifikacija na novim ulaznim podacima. Isto tako ako dođe do prekida prilikom učenja iz bilo kojeg razloga, mreža se može nastaviti trenirati od trenutka spremljenog stanja. Spremanje mreže se postiže pozivanjem funkcije `save` iz `tf.train.Saver` klase:

```
saver = tf.train.Saver()
saver.save(session, "models/model_lstm_best.ckpt")
```

Model se kasnije učitava naredbom *`restore`*:

```
saver.restore(session, model)
```

Prije učitavanja modela potrebno je inicijalizirati cijelu mrežu kao i prilikom treniranja, što je princip Tensorflow biblioteke – spremanje modela je zapravo spremanje naučenih utega (engl. *weights),* pragova (engl. *bias*) i ostalih parametara koji se mijenjaju i optimiziraju u procesu treniranja. Varijable, dimenzije i model mreže se inicijalizira prije učitavanja spremljene konfiguracije.

5. Analiza sentimenta na hrvatskom jeziku

Za razliku od engleskog, hrvatski jezik je morfološki bogat i složen za obradu. Uz to, ne postoji dovoljno velik skup podataka koji bi bio pogodan za treniranje u svrhu analize sentimenta. Iz tih razloga postoje drugi načini kako bi se riješio ovaj problem. U nastavku su izložena neka od istraživanja u ovom polju.

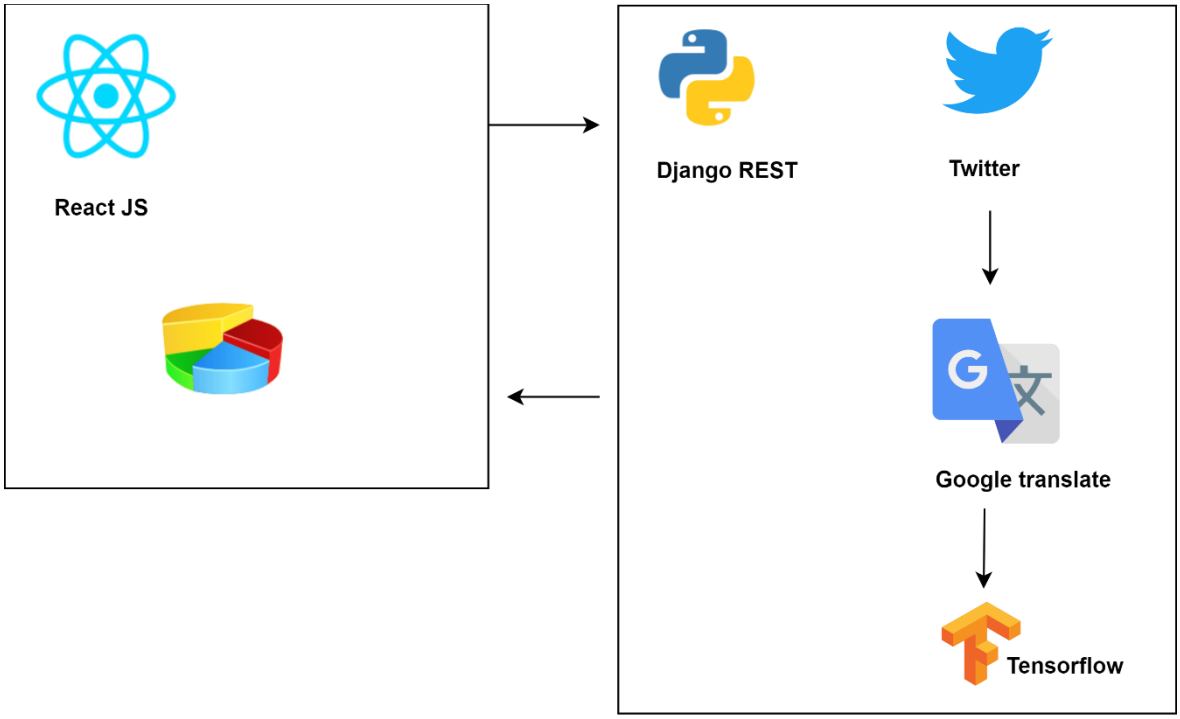
U radu „Comparison of Short-Text Sentiment Analysis Methods for Croatian“, (Rotim et al., 2017.) opisano je testiranje modela temeljenih na SVM-u na kratkim tekstovima. Podaci su skupljeni iz 3 skupa podataka na hrvatskom jeziku; „Game Reviews“ (GR) – skup od 1858 recenzija video igara, „Domain-specific tweets“ (TD) – skup od 2967 Twitter komentara (engl. *tweets*) povezanih uz emisiju „Glas Hrvatske“ i „General-topic tweets“ (TG) – skup od 7999 kratkih komentara raznolikih tema. Dobivena preciznost je na većini modela bila nešto više od 50%, a najbolji rezultat od 82% postignut je samo na jednom od 3 skupa, uz 71 i 54% na drugom i trećem.

U radu „Predicting Croatian Phrase Sentiment Using a Deep Matrix-Vector Model“ (Biđin et al., 2014.) korištena je tehnika rekurentne neuronske mreže. Skupovi podataka su bili „Synthetic dataset“ – umjetno kreiran skup od 1500 fraza hrvatskih riječi i „Movie review dataset“ – skup podataka prevedenih filmskih recenzija od 1026 fraza. Rezultati koji su izneseni potvrdili su učinkovitost dubokih neuronskih mreža na analizi sentimenta, posebice rekurentnih neuronskih mreža (RNN), što je i u ovom radu odabrano kao najbolji model (LSTM mreža spada pod RNN tip neuronske mreže).

U radu „Identifikacija online imidža organizacija temeljem analize sentimenata korisnički generiranog sadržaja na hrvatskim portalima“ (Jakopović et al., 2016.) , kao alat za analizu sentimenta izabran je besplatni program *SentiStrength* – program za analizu i obradu teksta na engleskom jeziku, s mogućnosti prilagodbe na druge jezike, uključujući hrvatski. Program omogućuje prepoznavanje polariteta (pozitivan i negativan sentiment) na ljestvici od 1-5. Prilagodba se odvija prevađanjem tekstova s engleskog na hrvatski, koristeći odabrani hrvatski leksikon i tako se stvara model za klasifikaciju. Iako nije optimalno, ovaj princip otvara alternativnu opciju rješavanja nedostatka podataka na hrvatskom jeziku.

Uz navedene prepreke, u ovom radu u svrhu web aplikacije odabran je sljedeći princip – analiza sentimenta odrađena je uz prevođenje teksta s hrvatskog na engleski, korištenje naučenog modela na engleskom jeziku, klasifikacija sentimenta i vraćanje rezultata.

6. Web aplikacija



Slika 10. Web aplikacija

Web aplikacija sastoji se od:

- React web aplikacije - klijentskog dijela (engl. *frontend*) razvijenog kao SPA (engl. *single page application*, skraćeno SPA)
- Django REST – poslužiteljskog dijela (engl. *backend*) razvijenog kao REST servis
- Python modula za analizu sentimenta
- Python modula za prikupljanje podataka s društvene mreže Twitter

Za razliku od tradicionalnih web aplikacija gdje se prilikom svakog poziva poslužitelju odgovara novom HTML stranicom (engl. *page refresh*), u slučaju jednostranične aplikacije (SPA) aplikacija se pokreće na prvo otvaranje stranice i svaki sljedeći poziv odvija se kroz Ajax (engl. *Asynchronous JavaScript and XML*) pozive, koji vraćaju rezultat bez potrebe za ponovnim učitavanjem, a rezultat je obično zapisan u JSON (engl. *Javascript Object Notation*) formatu. Aplikacijska logika prebačena je na klijentsku stranu, što omogućuje brze promjene stanja i responzivno sučelje. Poslužiteljski dio je odvojen i služi kao REST (engl. *Representational State Transfer*) servis čije su karakteristike:

- Komunikacija bez stanja (engl. *stateless*)
- Upotreba definiranih standardnih metoda – GET, POST, PUT, DELETE

- Svaki resurs ima jedinstveni identifikator URI (engl. *Uniform Resource Indentifier*, skraćeno URI)
- Klijent-server arhitektura, gdje više klijenata može koristiti isti centralni server
- Uniformno sučelje

6.1. Django REST

Django je Python web programski okvir koji podržava fleksibilnu arhitekturu i slaganje više modula i aplikacija u jednu cjelinu. Prilikom kreiranja Django projekta postoje glavne skripte:

- settings.py – konfiguracija projekta
- urls.py – konfiguracija dostupnih URL-ova
- wsgi.py – konfiguracija web servera

Kako bi se koristio REST programski okvir, potrebno ga je instalirati i dodati u *settings.py* skriptu. Također se registrira i aplikacija za klasifikaciju:

```
INSTALLED_APPS = [
    …
    'rest_framework',
    'sentiment.apps.ClassifierConfig'
]
```

U skripti *models.py* kreiraju se modeli koji su zapravo Python klase. Glavni model označava Twitter komentar (tweet):

```
class Tweet(models.Model):
    id = models.IntegerField(max_length=64, null=False)
    text = models.CharField(max_length=280, null=False)
    created_at = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey('auth.User')
    likes_count = models.IntegerField(default=0, null=False)
    comments_count = models.IntegerField(default=0,
null=False)
```

Kôd 8. Model Twitter poruke

Model se sastoji od sljedećih polja:

- id – identifikator tweeta
- text – tekst poruke, maksimalne dužine 280 znakova
- created_at – datum nastanka tweeta

- user – korisnik koji je poslao tweet
- likes_count – broj „lajkova“ na tweet
- comments_count – broj komentara na tweet

Za pretvaranje modela u JSON format pogodan za klijentsku stranu, koristi se serijalizacija preko Django Rest serijalizatora unutar *serializers.py* skripte:

```
from rest_framework import serializers
from .models import Tweet
class TweetSerializer(serializers.ModelSerializer):
    class Meta:
        model = Tweet
        fields = ('id', 'text', 'created_at', 'user',
'likes_count' 'comments_count')
```

Kôd 9. Serijalizacija

Dostupni URL-ovi konfiguriraju se u skripti *urls.py* dodavanjem u varijablu `urlpatterns` koja služi za mapiranje putanja odnosno pogleda (engl. *views*) i URL-ova:

```
from django.conf.urls import path
from sentiment.views import ClassifySentimentView,
TranslateTweetView, GetTweetsView
…
urlpatterns = [
…
path('api/classify', ClassifySentimentView.as_view()),
path('api/translate', TranslateTweetView.as_view()),
path('api/tweets', GetTweetsView.as_view()),
]
```

Kôd 10. REST mapiranje

Glavni servis je `api/classify` koji poziva funkciju `classify_sentiment` za klasifikaciju sentimenta koristeći model LSTM neuronske mreže iz prethodnih poglavlja:

```
from rest_framework.views import APIView
from sentiment.classifier import classify_sentiment
class ClassifySentimentView(APIView):
    def post(self, request):
        …
        analyzed_tweets = {}
        for tweet in tweets:
            analyzed_tweets.update({tweet.id:
classify_sentiment(tweet)})
```

```
        return JsonResponse({'data':analyzed_tweets})
```

Kôd 11. Klasifikacija tweetova

Za prijevod teksta s hrvatskog na engleski jezik, koristi se biblioteka `googletrans`.

```
from googletrans import Translator
def translate_cro_to_eng(tweets):
    translator = Translator()
    translations = translator.translate(tweets, dest='en',
src='hr')
    return translations
```

Kôd 12. Prijevod teksta

Funkcija `translate` prima listu tweetova, parametar `dest` označava ciljani jezik, a parametar `src` izvorni jezik. Dobiva se lista s prevedenim ulaznim podacima. Ova funkcija se tako koristi u servisu `api/translate` kako bi se prije klasifikacije sentimenta tweetovi preveli na engleski jezik.

Servis `api/tweets` koristi se za dohvaćanje tweetova koristeći Twitter API.

Django web server se pokreće naredbom `runserver` iz *manage.py* skripte.

6.2. Twitter API

Za dohvaćanje podataka s društvene mreže Twitter, koristi se Twitter API, koji je dostupan preko Python biblioteke Tweepy. Potrebno je registrirati Twitter razvojni račun (engl. *developer account*) kako bi se moglo autenticirati. Autentikacija se odvija na sljedeći način:

```
import tweepy
…
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
```

Pomoću `OAuthHandler` klase postavljaju se potrebni parametri za autentikaciju i postavlja se pristupni token. Nakon toga moguće je koristiti niz dostupnih funkcija iz Tweepy biblioteke, od koji je u ovom projektu najbitnija funkcija za pretraživanje tweetova:

```
api = tweepy.API(auth)
…
tweets = api.search(q="Algebra", lang="hrv", rpp=10):
```

Funkcija `search` prima parametar `q` koji označava traženi pojam, `lang` koji označava jezik pretrage i `rpp` koji označava maksimalni broj vraćenih tweetova.

Pretraživanje je moguće i preko korisničko imena pozivom funkcije `user_timeline`.

```
tweets = api.user_timeline(screen_name=user, count=10)
```

Tweepy biblioteka omogućuje protok podataka u realnom vremenu (engl. *streaming*). Instanca klase `tweepy.Stream` omogućuje otvaranje sesije (engl. *session*) i slanje poruka pomoću `StreamListener` klase. Funkcija `on_data` sluša poruke i poziva funkcije ovisno o tipu poruke. Proces se sastoji od tri koraka:

- Kreiranje klase koja nasljeđuje `StreamListener`
- Kreiranje `Stream` objekta pomoću navedene klase
- Spajanje na Twitter API koristeći `Stream`

Učitavaju se potrebne klase iz tweepy biblioteke:

```
from tweepy import Stream
from tweepy.streaming import StreamListener
```

Kreira se klasa `TwitterListener` klasa koja koristi `StreamListener`:

```
class TwitterListener(StreamListener):
  def __init__(self, pipe_func):
    self.pipe_func = pipe_func
...
  def on_data(self, data):
    if (self. pipe_func):
      self.pipe_func(data)
    return True
...
  def on_error(self, status):
    if (status == 420):
      return False
```

Kôd 13. Twitter Listener

Klasa `TwitterListener` nasljeđuje dvije funkcije iz klase `StreamListener`, a to su `on_data` koja se poziva nakon dohvaćanja tweetova i `on_error` koja se okida u slučaju iznimke. *Streaming* objekt se kreira koristeći navedenu `TwitterListener` klasu:

```
twitter_listener = TwitterListener(filter_func)
twitter_stream = tweepy.Stream(auth = api.auth, listener=
twitter_listener ())
```

6.3. Django Channels

Channels je projekt koji proširuje Django u pogledu otvorenih konekcija, *WebSocketa*, IoT protokola i slično. Temelji se na Python specifikaciji ASGI (ASGI, 2018.).
Pomoću *WebSocketa* uspostavlja se komunikacija između servera i klijenta tako da server može slati poruke klijentima i uzrokovati promjenu stanja aplikacije bez ponovnog učitavanja stranice. Protok podataka odvija se u realnom vremenu. Za razliku od HTTP protokola, gdje samo klijent koji je zatražio resurs dobiva rezultat, pomoću *WebSocketa* server može poslati poruku više klijenata odjednom. *WebSocket* poruke se zovu s prefiksom `ws://` umjesto `http://`.
Kreira se klasa `TwitterConsumer` koja nasljeđuje iz klase `JsonWebsocketConsumer`:

```
class TwitterConsumer(JsonWebsocketConsumer)
    def connect(self):
       self.send({"type": "websocket.accept"})
    def disconnect(self, code):
       self.send({"type":"websocket.disconnect",
        "code":code})
```

Ova klasa implementira funkciju `connect` preko koje se dodaje u listu klijenata i počinje slušati poruke od servera. Isto tako implementira i funkciju `disconnect` preko koje se isključuje. Instanca `TwitterConsumer`, odnosno *Consumer* je u paraleli isto što i pogled (engl. *view*) u Django REST-u, pa se mapira na željenu rutu.
U skripti *routing.py* definira se ruta do servisa pozivom klase `ProtocolTypeRouter`:

```
from tweet.consumers import TwitterConsumer
from channels.routing import ProtocolTypeRouter, URLRouter
...
channel_routing = ProtocolTypeRouter({
    "websocket": AuthMiddlewareStack(
        URLRouter([
            url(r'api/twitter-stream/', TwitterConsumer),
        ]),
    ),
})
```

Kôd 14. Channel Routing

6.4. React JS

React JS je moderna JavaScript biblioteka za izradu korisničkog sučelja. Glavne karakteristike su:

- komponente – modularnost komponenti omogućuje slaganje odvojenih logičkih cjelina i održivost kôda
- jednosmjerni protok podataka (engl. *one way data binding*)
- JSX (JavaScript XML) - ekstenzija JavaScripta koja u kombinaciji s HTML-om pruža jednostavniji način izrade sučelja
- virtualni DOM – reprezentacija originalnog DOM objekta, koja se mijenja ovisno o stanju aplikacije. Nakon promjene se uspoređuje virtualni i pravi DOM, i osvježava se samo dio sučelja koji se promijenio, a ne cijela stranica. To ubrzava rad aplikacije i smanjuje nepotrebno trošenje memorije

Komponente u Reactu mogu biti JavaScript klase ili funkcije. Postoji hijerarhija komponenti što znači da postoje komponente roditelji (engl. *parent*) i djeca (engl. *child*). Tok podataka uvijek ide od vrha prema dnu hijerarhije. *Parent* komponenta prosljeđuje *child* komponenti podatke preko parametara koji se zovu `props`. Ovo omogućuje dinamičko prikazivanje *child* komponenti ovisno o parametrima koji su poslani. Glavna i jedina nužna funkcija u komponenti je `render()` koja vraća sadržaj komponente koji će se prikazati na ekranu.

Dio komponente koja prikazuje Tweet izgleda ovako:

```
export default class TweetWrapper extends React.Component {
  const item = {props.tweetData}
  render () {
    return (
      <Card>
       <CardHeader>
         <img src={item.profile_image_url} ></img>
         <p className={classes.cardCategory}}>
           {item.screen_name}</p>
         <p className={classes.cardCategory} >{item.text}</p>
       </CardHeader>
    </Card>    )
  }
}
```

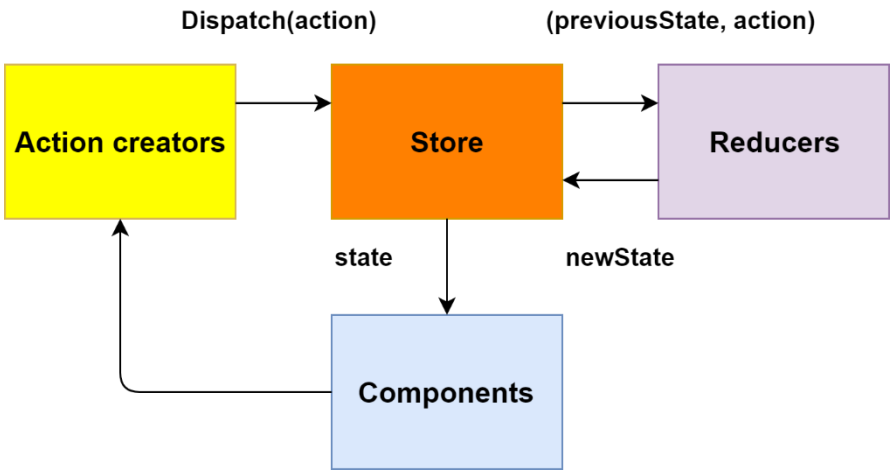
Kôd 15. Tweet komponenta

`TweetWrapper` komponenta se koristi kako bi se prikazala lista dohvaćenih tweetova. Ta lista se dijeli na dio s pozitivno i dio s negativno klasificiranim tweetovima. Isto tako prikazuje se i statistika koliko ih je pozitivno, a koliko negativno klasificirano.

Za kontrolu nad stanjem podataka u aplikaciji koristi se biblioteka Redux. Sastoji se od sljedećih elemenata:

- spremište stanja (engl. *store*) – čuva stanje aplikacija na jednom mjestu
- akcije – prenose informacije da se nešto dogodilo i javljaju da je potrebno reagirati na događaj
- reduktori – definiraju što se treba dogoditi na pojedinu akciju. To su čiste funkcije koje primaju prethodno stanje aplikacije i iz njih kreiraju novo stanje. Za isti ulaz uvijek daju isti izlaz.

Protok podataka vidljiv je na sljedećoj slici:



Slika 11. Redux protok podataka

Iz komponente počinje protok podataka okidanjem korisničke akcije. Definirana Redux akcija šalje zahtjev za promjenu stanja, koju obrađuje reduktor namijenjen za tu akciju. U spremištu se postavlja novo stanje koje okida novi poziv `render` metode u komponentama koje koriste dio stanja aplikacije koji se promijenio.

Za komunikaciju između klijentske i serverske aplikacije koristi se biblioteka axios koja koristi JavaScript Promise, mehanizam asinkronog pozivanja funkcija. Poziv za klasificiranje sentimenta šalje listu tweetova na obradu, i nakon povratnog rezultata okida funkciju `dispatch` koja javlja da je potrebno promijeniti stanje i osvježiti komponente:

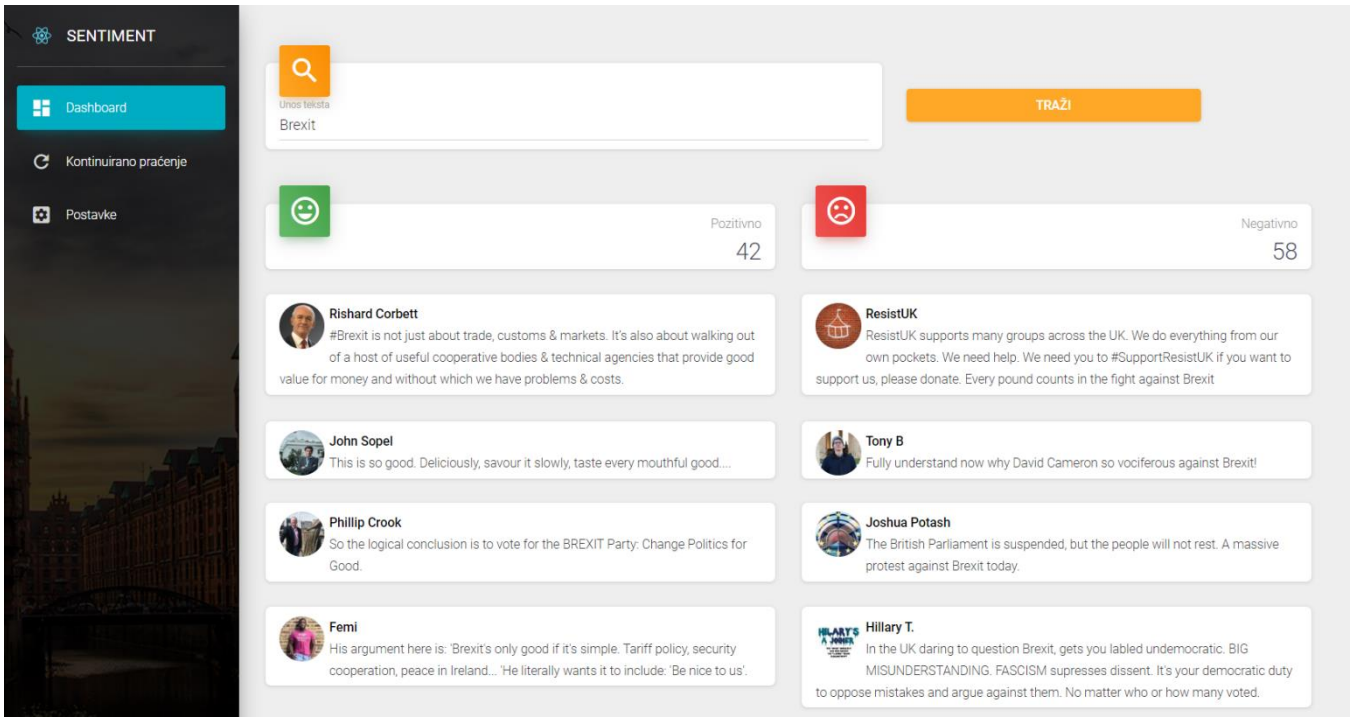
```
axios.post(config.baseUrl + "/api/classify",
{tweets:tweetData})
```

```
.then(response => {
  dispatch(setTweetsSentiment(response.data))
})
.catch(error => {
  throw(error);
});
```

Kôd 16. Asinkroni poziv za klasificiranje sentimenta

Kôd 16. prikazuje korištenje axios biblioteke za pozivanje Django REST servisa.

6.5. Sučelje



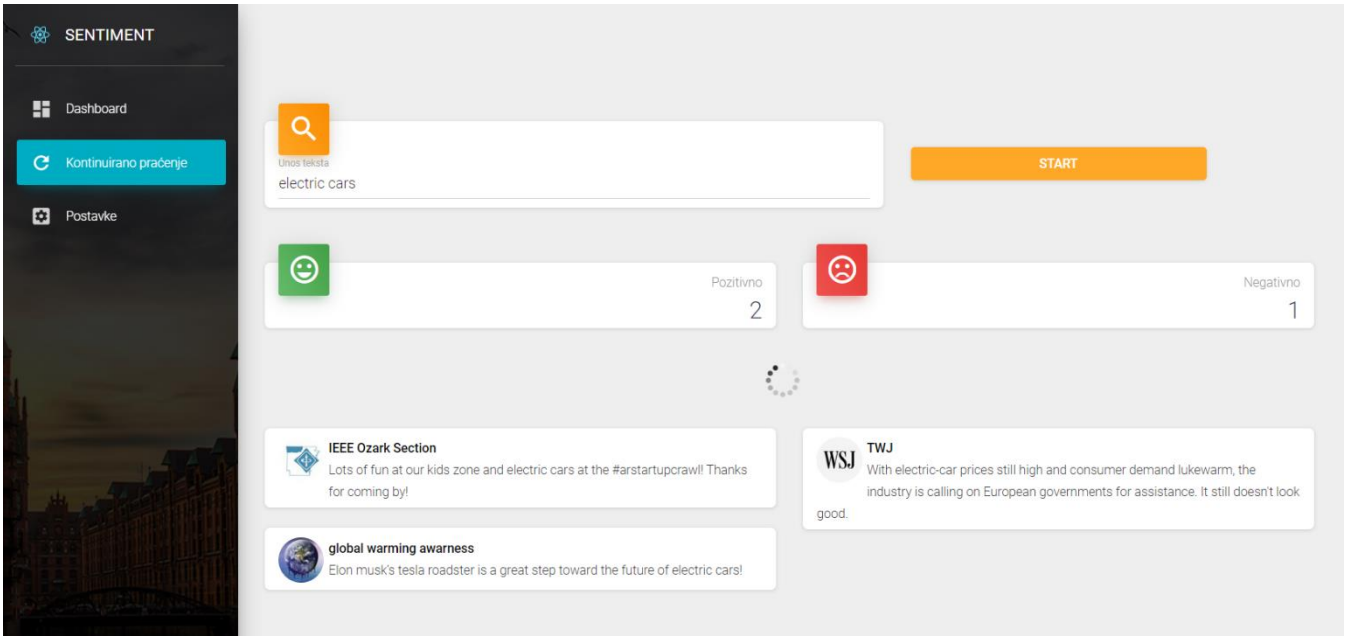
Slika 12. Sučelje web aplikacije

Sučelje web aplikacije sastoji se od dva dijela:

- izbornik s lijeve strane
- glavni dio sadržaja s desne strane

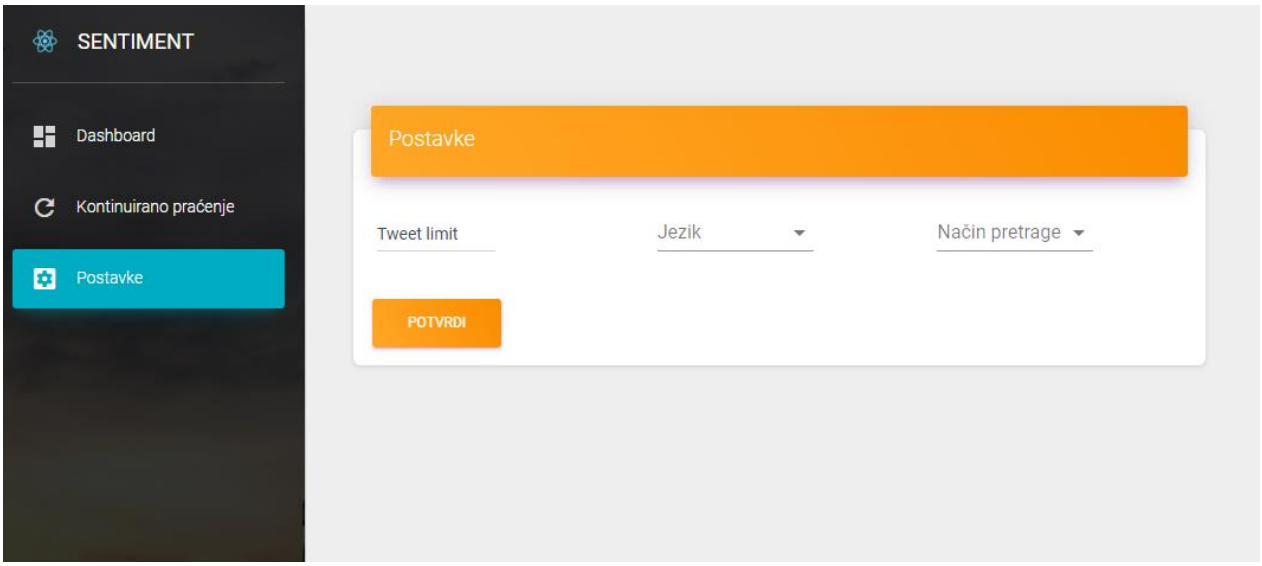
Izbornik sadrži opcije „Dashboard, „Kontinuirano praćenje“ i „Postavke“. „Dashboard“ je

stranica koja se sastoji od polja za unos ključne riječi i gumba za pretraživanje. Na pritisak gumba „Traži“ pretražuju se tweetovi i ovisno o rezultatu analize sentimenta prikazuju se u lijevom ili desnom stupcu.



Slika 13. Kontinuirano praćenje

„Kontinuirano praćenje“ je stranica koja u realnom vremenu kontinuirano osvježava sadržaj stranice. Kao i kod stranice „Dashboard“ postoji polje za unos riječi i gumb za pokretanje. Na pritisak gumba „Start“ pokreće se komunikacija između Twitter *streaming* API-ja i klijentske aplikacije. Novi tweetovi se dohvaćaju u realnom vremenu, klasificiraju i prikazuju u lijevom ili desnom stupcu ovisi o rezultatu analize sentimenta.

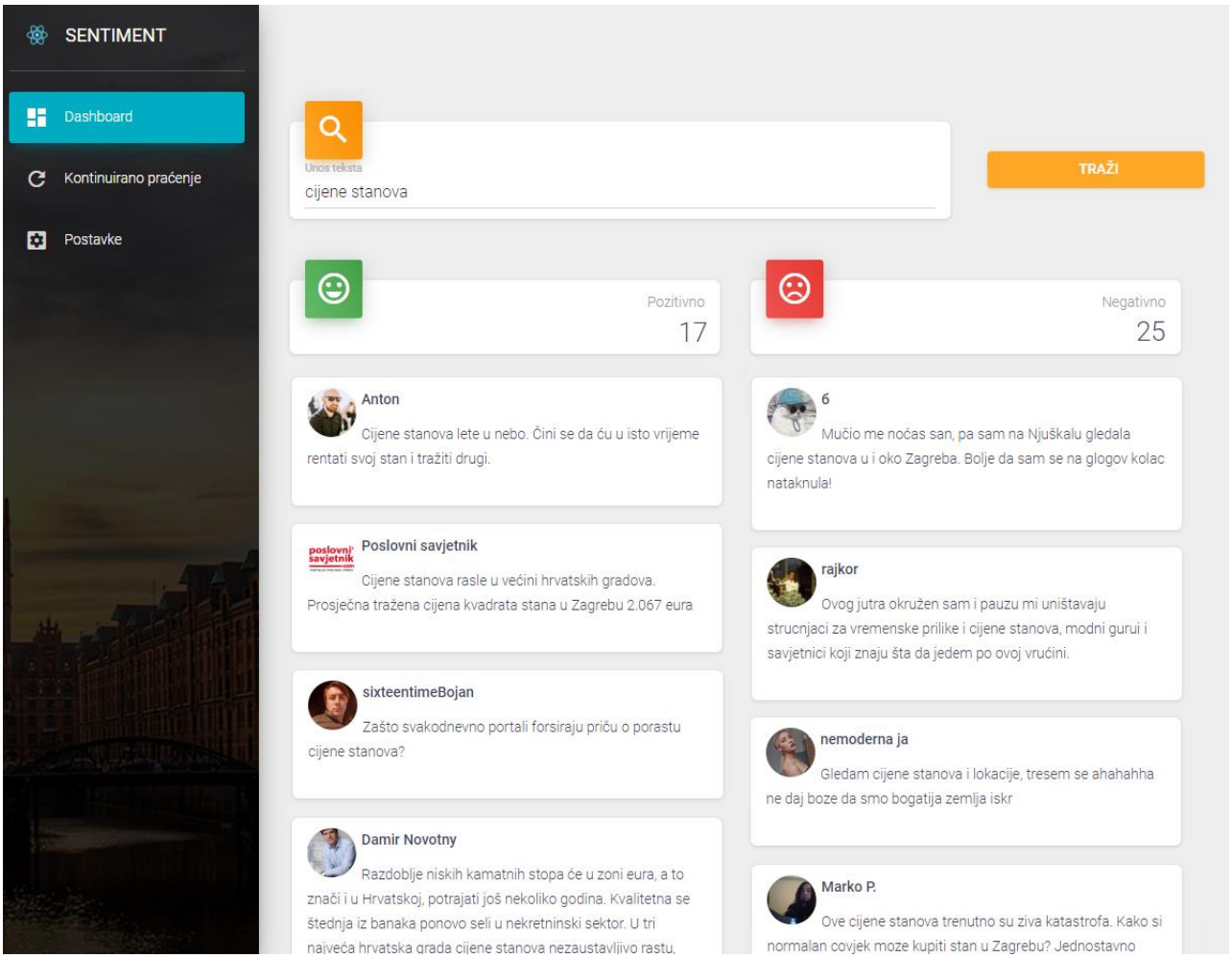


Slika 14. Postavke web aplikacije

Slika 14. prikazuje stranicu postavki web aplikacije sa sljedećim opcijama:

- Tweet limit - maksimalni broj tweetova koji će se dohvatit prilikom pretraživanja
- Jezik - odabir između hrvatskog ili engleskog jezika
- Način pretrage - pretraživanje prema ključnoj riječi ili prema korisničkom imenu

Na Slika 15. može se vidjeti primjer rezultata aplikacije ako je u postavkama odabran hrvatski jezik i način pretrage po ključnoj riječi „cijene stanova“.



Slika 15. Pretraživanje na hrvatskom jeziku

Zaključak

Komentari korisnika na web portalima, društvenim mrežama i blogovima prikazuju stav i povratnu informaciju nad aktualnima događajima i trendovima. Može se pretpostaviti da su korisnici slobodniji izraziti se tim putem nego anketama i intervjuima. Iz toga proizlazi prilika za automatizaciju prikupljanja takvih podataka u svrhu analize i otkrivanja znanja, a jedna od metoda je i analiza sentimenta, odnosno osjećaja i stavova publike o nekoj temi.

Ovakva analiza posebno je bitna na područja marketinga, prodaje, odnosa sa javnošću, pa čak i financijskog sektora i predviđanja vrijednosti dionica.

Koristeći programski jezik Python i biblioteke za strojno učenje, prikazana je implementacija različitih algoritama za analizu sentimenta. Na IMDb skupu podataka od 50,000 recenzija filmova, najbolje rezultate postigla je LSTM neuronska mreža s preciznosti od 93.44%. Za postizanje visoke preciznosti potrebno je skupiti što veći broj uzoraka za treniranje, što je još uvijek problem u slučaju hrvatskog, a i drugih morfološki bogatih jezika. Iz tog razloga, odabran je alternativni način rješenja primjenom prevođenja hrvatskog teksta na engleski jezik, nad kojim je moguće provesti analizu sentimenta s visokom preciznosti i vratiti rezultate. Izrađena je web aplikacija koristeći Django web programski okvir i React JS biblioteku za izradu korisničkog sučelja. Proces se sastoji od pretraživanja društvene mreže Twitter i sakupljanja tweetova, koji se obrađuju i pomoću spomenute neuronske mreže klasificiraju kao pozitivni ili negativni.

Popis kratica

API	*Application Programming Interface*	aplikacijsko programsko sučelje
ASGI	*Asynchronous Server Gateway Interface*	sučelje za asinkronu komunikaciju
AWS	*Amazon web services*	Amazon web servisi
HTML	*Hypertext Markup Language*	prezentacijski jezik za izradu web stranica
IoT	*Internet of Things*	Internet stvari
LSTM	*Long-short term network*	mreža s kratkotrajnim pamćenjem
NLP	*Natural Language Processing*	obrada prirodnog jezika
RNN	Recurrent Neural Network	rekurentna neuronska mreža
SVM	*Support vector machine*	model potpornih vektora
URL	*Uniform Resource Locator*	lokator resursa ili sadržaja na internetu

Popis slika

Slika 1. SVM prikaz razdvajanja skupa podataka 7

Slika 2. SVM prikaz regularizacije 7

Slika 3. Umjetni neuron 8

Slika 4. Duboka neuronska mreža 9

Slika 5. Pretreniranje 9

Slika 6. LSTM neuronska mreža 10

Slika 7. Vektorski prikaz riječi 19

Slika 8. Arhitektura duboke neuronske mreže 20

Slika 9. LSTM neuronska mreža 21

Slika 10. Web aplikacija 24

Slika 11. Redux protok podataka 31

Slika 12. Sučelje web aplikacije 32

Slika 13. Kontinuirano praćenje 33

Slika 14. Postavke web aplikacije 33

Slika 15. Pretraživanje na hrvatskom jeziku 34

Popis tablica

Tablica 1. Leksikon 4

Tablica 2. IMDb skup podataka 12

Tablica 3. Usporedba algoritama 22

Popis kôdova

Kôd 1. Priprema podataka 13

Kôd 2. Leksikon 15

Kôd 3. Naivni Bayesov klasifikator 16

Kôd 4. Algoritam maksimalne entropije 17

Kôd 5. SVM 18

Kôd 6. Duboka Neuronska mreža 20

Kôd 7. LSTM neuronska mreža 21

Kôd 8. Model Twitter poruke 25

Kôd 9. Serijalizacija 26

Kôd 10. REST mapiranje 26

Kôd 11. Klasifikacija tweetova 27

Kôd 12. Prijevod teksta 27

Kôd 13. Twitter Listener 28

Kôd 14. Channel Routing 29

Kôd 15. Tweet komponenta 30

Kôd 16. Asinkroni poziv za klasificiranje sentimenta 32

Literatura

[1] BO P., LILLIAN L., *Opinion Mining and Sentiment Analysis, Foundations and Trends in Information Retrieval,* 2008.

[2] GLAVAŠ, G., ŠNAJDER, J., BAŠIĆ, B. *Semi-Supervised Acquisition of Croatian Sentiment Lexicon,* 2012

[3] SHUOHANG WANG, JING JIANG, *Learning Natural Language Inference with LSTM, Singapore Management University,* 2016.

[4] GO, AL., BHAYANI, R., LEI HUANG, *Twitter Sentiment Classification using Distant Supervision,* 2009.

[5] MOLLA, RANI. *Web stranica*. 2012.

https://contently.com/2012/10/24/social-media-sentiment-becomes-factor-in-presidential-campaigns/

[6] ZHANG, Y., JIN, R., ZHOU, Z., *Understanding Bag-of-Words Model: A Statistical Framework,* 2010.

[7] GATTI, L., GUERINI, M., & TURCHI, M., *SentiWords: Deriving a high precision and high coverage lexicon for sentiment analysis,* 2016.

[8] SENTILEX, *Web stranica,* 2012.

http://takelab.fer.hr/sentilex

[9] MUSTO, C., SEMERARO, G., POLIGNANO, M. *A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts,* 2014.

[10] RAO, Y. LI, J., XIANG, X., XIE, H. , *Intensive Maximum Entropy Modelfor Sentiment Classification of Short Text, 2015.*

[11] DINU, L., IUGA, I. *The Naive Bayes Classifier in Opinion Mining: In Search of the Best Feature Set,* 2012.

[12] PANG, B., LEE, L., *Thumbs up? Sentiment Classification using Machine Learning Techniques,* 2002.

[13] MOUNT, J. *The equivalence of logistic regression and maximum entropy models,* 2011.

[14] ASGI, *Web stranica,* 2018.

https://asgi.readthedocs.io/en/latest/

[15] MEDIUM, *SVM: (Support Vector Machine) – Theory, Web stranica,* 2017.

https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72

[16] BAŠIĆ, B.D., ĆUPIĆ, M., ŠNAJDER, J. *Umjetne neuronske mreže*, Zagreb 2008.

[17] SOCHER, R., PERELYGIN, A., WU, J. *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treeban*k, 2013.

[18] MAAS, A., DALY, R. *Learning Word Vectors for Sentiment Analysis,* 2011.

[19] ROTIM, L., ŠNAJDER, J., *Comparison of Short-Text Sentiment Analysis Methods for Croatian*, 2017.

[20] AGIĆ, Ž.,LJUBEŠIĆ, N., TADIĆ, M., *Predicting Croatian Phrase Sentiment Usinga Deep Matrix-Vector Model,*

[21] FREITAS, N. Machine Learning-Lectures Oxford, 2014.

https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/ kolovoz 2016.

[22] *Understanding LSTM Networks*, *Web* STRANICA, 2015.

http://colah.github.io/posts/2015-08-Understanding-LSTMs

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu,_____.

Prilog

Uz ovaj rad, priložen je CD koji sadrži izvorni kod opisane aplikacije u Pythonu i Reactu uz potrebnu dokumentaciju.

Slike u ovom radu koje nisu referencirane u fusnotama, izrađene su koristeći alate lucidchart.com i draw.io. Slike i pripadajuće .xml datoteke priložene su na navedenom CD-u.

Algebra
visoka škola za
primijenjeno računarstvo

Web aplikacija za analizu sentimenta na društvenim mrežama

Pristupnik: Denis Košutić, 0036449738

Mentor: v.pred. Aleksander Radovan, dipl. ing.