ZAHVALA

Zahvaljujem se mentoru prof. dr. sc. Domagoju Jakoboviću na vodstvu, razumijevanju, savjetovanju i pomoći pri izradi završnog rada.

Zahvaljujem roditeljima i mom starijem bratu na neprekidnoj podršci tijekom mojeg obrazovanja.

Sadržaj

Uvod

Strojno učenje je područje kojem popularnost raste iz dana u dan. Svrha strojnog učenja je nalaženje pravilnosti u podacima i ekstrakcija informacije iz mnoštva strukturiranih, pa tako i nestrukturiranih podataka.

Analiza sentimenta je jedna od zanimljivijih primjena strojnog učenja, gdje se obradom prirodnog jezika (engl. *natural language processing*) i primjenom nekih algoritama strojnog učenja analiziraju emocije unutar tekstualnih podataka. Analiza sentimenta postaje sve više relevantnija kako ljudi sve otvorenije izražavaju svoja mišljenja i emocije, da bi se kroz tekstualan zapis tih mišljenja izvukla korisna informacija kao što je opće mišljenje o nekom brandu ili proizvodu.

Analizom sentimenta brand može uočiti pod kojim uvjetima se sentiment na određene načine mijenja, kad pada i kad raste, što omogućuje bolje prilagođavanje usluga i proizvoda prema korisnicima.

Na primjer, analiziranje sentimenta nad 4000+ rječitih odgovora uzvraćenim u anketama zadovoljstva koje ispunjavaju potrošači moglo bi pomoći u otkrivanju onoga što potrošače čini sretnima ili nesretnima, u svakoj fazi kroz koju potrošač prolazi koristeći usluge ili proizvode nekog branda [1].

U ovom radu opisana je primjena analize sentimenta nad recenzijama filmova, koje su također tekstovi s izraženim mišljenjima, gdje su mišljenja upućena pojedinom filmu.

Opisana je primjena analize sentimenta nad skupom recenzija s IMDB platforme, i to pomoću algoritma neuroevolucije koji kombinira neuronske mreže i evolucijom inspirirane algoritme.

Cilj ovog rada je istraživanje ponašanja i mogućnosti algoritma neuroevolucije u svrhu precizne analize sentimenta.

1. Ulazni skup podataka

Dobavljeni skup podataka za učenje ovog algoritma sadrži 50000 recenzija filmova s IMDB platforme, od kojih je uzet podskup od 540 recenzija u svrhu demonstracije algoritma.

1.1. IMDB recenzije filmova

Dobavljeni skup od 50000 IMDB recenzija filmova podijeljen je na podjednake skupove od 25000 recenzija za učenje i 25000 recenzija za testiranje. Raspodjela recenzija po sentimentu je binarna i uravnotežena, tj. 25000 pozitivnih recenzija i 25000 negativnih recenzija. Svaka recenzija označena je pozitivnim ili negativnim sentimentom. Negativnim sentimentom smatraju se recenzije filmova s ocjenom 4/10 naniže, dok se pozitivnim sentimentom smatraju recenzije filmova s ocjenom 7/10 naviše, što čini ovaj skup podataka prilično polarnim te nema neutralnih recenzija.

Recenzije su unaprijed obrađene i zapisane kao nizovi cjelobrojnih indeksa riječi, gdje pojedini indeks označava učestalost odgovarajuće riječi u cijelom skupu recenzija (što je pogodno za operacije tipa „Nađi 1000 najučestalijih riječi“).

1.2. Pripravljanje podataka za učenje i testiranje

U ovom projektu, omjer podataka za učenje i testiranje je 7:3, tj. 70% za učenje i 30% za testiranje.

Recenzije su podrezane (engl. *truncated*) na maksimalnu duljinu od 500 riječi, tako da budu kompatibilne za daljnju obradu algoritmom, tj. tako da sve recenzije budu sličnog oblika prilikom dovođenja na ulaz neuronske mreže. Recenzije kraće od 500 riječi dodatno se pune vrijednošću 0 u indeksnoj reprezentaciji recenzija, gdje se nule umeću na početak liste indeksa (svejedno je da li se nule umeću na početak ili kraj, jer se riječi s indeksom 0 ne uzimaju u obzir prilikom određivanja sentimenta i u ovom kontekstu samo služe za punjenje do 500 riječi).

Učitane su reprezentacije od 5000 najučestalijih riječi iz skupa podataka, dok se ostale riječi smatraju nepoznatima i ne uzimaju se u obzir prilikom određivanja sentimenta.

Rječnik s preko 20000 riječi koje se javljaju u recenzijama filmova iz IMDB-ovog skupa već je pripremljen uz podatke te poredan po učestalosti riječi nad cijelim skupom podataka.

Podaci su dostupni na [2] te su uvedeni kroz sljedeći istraživački uradak: [3].

2. Struktura algoritma

Algoritam se sastoji od neuronske mreže i evolucijskog algoritma koji se primjenjuje nad neuronskom mrežom.

Težinske vrijednosti u neuronskoj mreži su realne vrijednosti u rangu od 0 do 1, uključivši. Te vrijednosti zajedno s aktivacijskim funkcijama čine strukturu neuronske mreže.

Svaka konfiguracija težinskih vrijednosti neuronske mreže predstavlja po jednu jedinku populacije. Jedinka populacije može se zapisati kao vektor (tj. lista) težinskih vrijednosti neuronske mreže. Kada se neka jedinka iz populacije evaluira, potrebno je vektor te jedinke pretvoriti u format težinskih vrijednosti koje neuronska mreža očekuje (vrijednosti organizirane po slojevima neuronske mreže).

Funkcija dobrote (engl. *fitness*) jedinki računa se pomoću evolucijskog algoritma nad jedinkama u populaciji, te se ovisno o njoj najbolje jedinke propagiraju u sljedeću iteraciju (tj. generaciju) algoritma, uz mutiranje i križanje jedinki.

2.1. Struktura neuronske mreže

Neuronska mreža sastoji se od 3 sloja: *Embedding* sloj kao prvi sloj, LSTM sloj kao središnji, te *Dense* sloj na izlazu. Za izradu neuronske mreže koristila se Keras biblioteka u Pythonu.

Konkretni podaci o strukturi neuronske mreže:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 500, 32)           160000
 lstm (LSTM)                 (None, 100)               53200
 dense (Dense)               (None, 1)                 101
=================================================================
Total params: 213,301
Trainable params: 213,301
Non-trainable params: 0
_________________________________________________________________
```

2.1.1. *Embedding* sloj

„*Word embeddings*“ označava familiju NLP tehnika koje preslikavaju semantičko značenje riječi u geometrijski prostor. To se postiže pridruživanjem brojčanog vektora svakoj riječi u definiranom rječniku, tako da „udaljenost“ između bilo koja dva vektora reprezentira semantičku vezu između dvaju riječi kojima su ti vektori pridruženi.

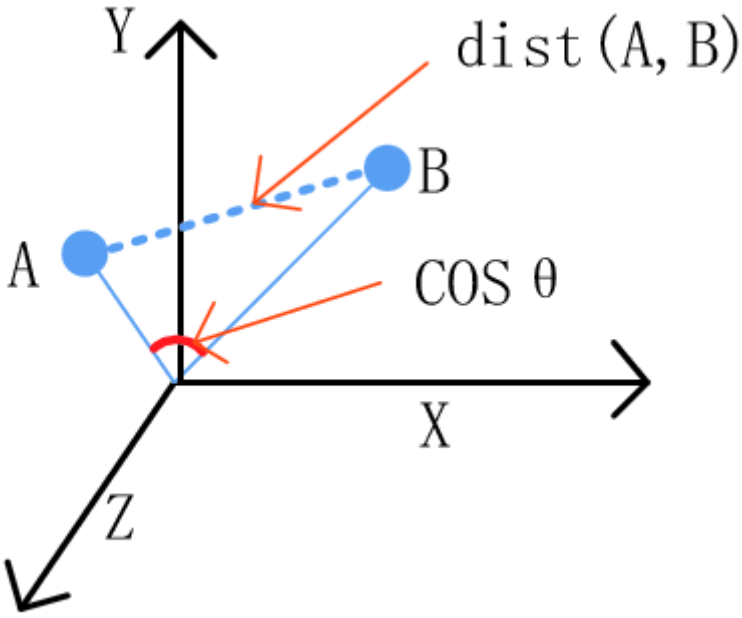
Udaljenost između vektora mjeri se kao Euklidova udaljenost ili češće kao „kosinusna sličnost“ (Slika 2.1).

Euklidova udaljenost u višedimenzionalnom prostoru računa se kao:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}$$

gdje su p_i i q_i koordinate riječi p i q u i-toj dimenziji.

Geometrijski prostor razapet tim vektorima zove se „*embedding* prostor“ (engl. *embedding space*).



Slika 2.1 Kosinusna sličnost između vektora A i B [4]

Na primjer, riječi „kokos“ i „lav“ su semantički vrlo različite, pa bi ih učinkoviti *embedding* prostor reprezentirao kao vektore koji su udaljeni jedan od drugog.

S druge strane, riječi „škola“ i „učenje“ su semantički povezane riječi, pa se trebaju nalaziti bliže u *embedding* prostoru.

U idealnom *embedding* prostoru, semantička veza između dvije semantički slične riječi kao „škola“ i „učenje“ se prikazuje vjerodostojno i precizno [5].

Embedding sloj u Keras API (kao i u većini alatnih okruženja za duboko učenje) vrši pretragu – za dani indeks riječi, vraća se gusto ispunjen *embedding* vektor.

Stoga, ima smisla zapisivati ulazni tekst kao listu indeksa pridruženih riječima, gdje je indeks za svaku riječ jedinstven cijeli broj.

Embedding tehnika predstavlja poboljšanje u odnosu na tradicionalni model torba-riječi (engl. *bag-of-word model*), gdje se koriste veliki, slabo ispunjeni vektori kako bi se reprezentirale pojedine riječi s puno vrijednosti parametara jednakim nula.

S druge strane, u *embedding* prostoru, riječi se prikazuju kao gusti vektori, gdje pojedini vektor prikazuje projekciju riječi na kontinuirani vektorski prostor.

U dubokom učenju, pozicija *embedding* vektora pojedine riječi unutar vektorskog prostora obično se uči ovisno o tome koje riječi se javljaju blizu navedene riječi u ulaznim tekstovima. Pozicija riječi u vektorskom prostoru obično se naziva *embedding*-om te riječi.

Dvije popularne metode za učenje *embedding*-a nad tekstom su:

- Word2Vec
- Glove

Duboko učenje se također može koristiti za učenje *embedding*-a [6].

U okviru ovog rada, za optimiziranje parametara neuronske mreže, kao i učenje parametara *embedding* sloja, koristi se algoritam neuroevolucije, to jest primjene evolucijskog algoritma u svrhu optimiranja neuronske mreže.

Embedding sloj je na početku inicijaliziran na pseudoslučajne težinske vrijednosti.

Embedding vektori se pomoću neuroevolucije formiraju nad ulaznim podacima za učenje.

Pri konstrukciji *Embedding* sloja u Keras-u, navode se tri parametra:

- **dimenzija ulaza**: veličina rječnika, broj poznatih riječi
  - odabrano je 5000 najčešćih riječi, stoga je dimenzija ulaza 5000
- **dimenzija izlaza**: veličina *embedding* vektora od kojih je *Embedding* sloj sačinjen
  - odabrana je vrijednost 32 kao veličina *embedding* vektora dovoljne ekspresivnosti
- **duljina ulaznih nizova**
  - postavljena je na 500 (riječi)

2.1.2. LSTM sloj

LSTM (engl. *Long Short-Term Memory*) je vrsta neuronske mreže koja za razliku od standardnih unaprijednih (engl. *feedforward*) neuronskih mreža ima povratne veze.

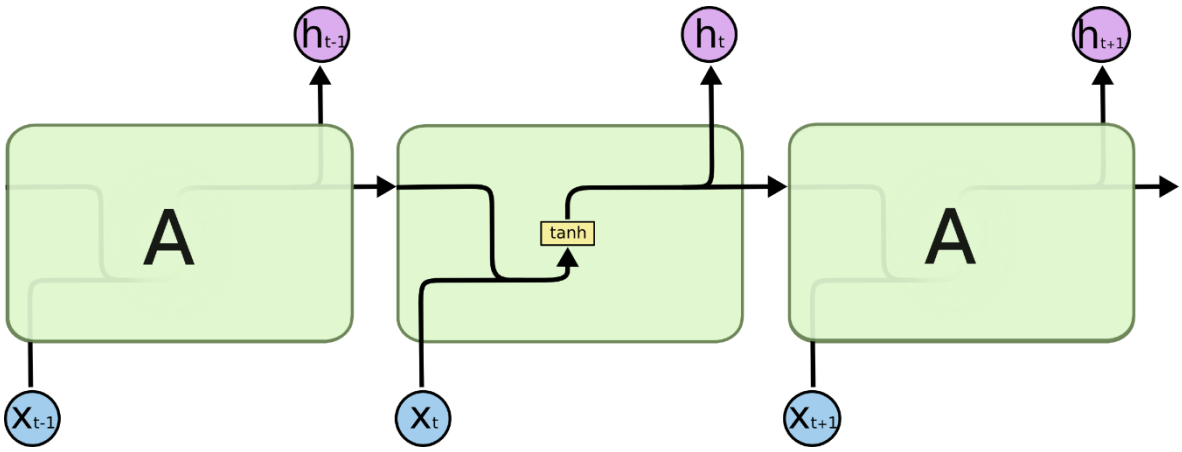
LSTM je podvrsta povratne neuronske mreže (engl. *recurrent neural network*), koja umjesto singularnih podatkovnih cjelina (npr. slike) može procesirati cijele nizove podataka (npr. ljudski govor i video).

LSTM mreže su prikladne za klasifikaciju, obradu i predviđanje nad podacima povezanim u diskretnom vremenu, jer vremenski razmaci između važnih događaja i podataka u vremenu mogu biti nepoznatog trajanja. LSTM mreže su napravljene za suočavanje s problemom nestajućeg gradijenta (engl. *vanishing gradient problem*) prisutnim u RNN, gdje unazadnom propagacijom (engl. *backpropagation*) gradijent može postajati sve manji i manji, do točke gdje se neuronska mreža više ne može učenje jer se težine efektivno više ne mogu mijenjati.

Još jedna prednost LSTM arhitekture nad RNN arhitekturom je relativna neosjetljivost na duljinu vremenskog razmaka između događaja, što je moguće zbog memorijskih ćelija sadržanim u LSTM modulima [7].

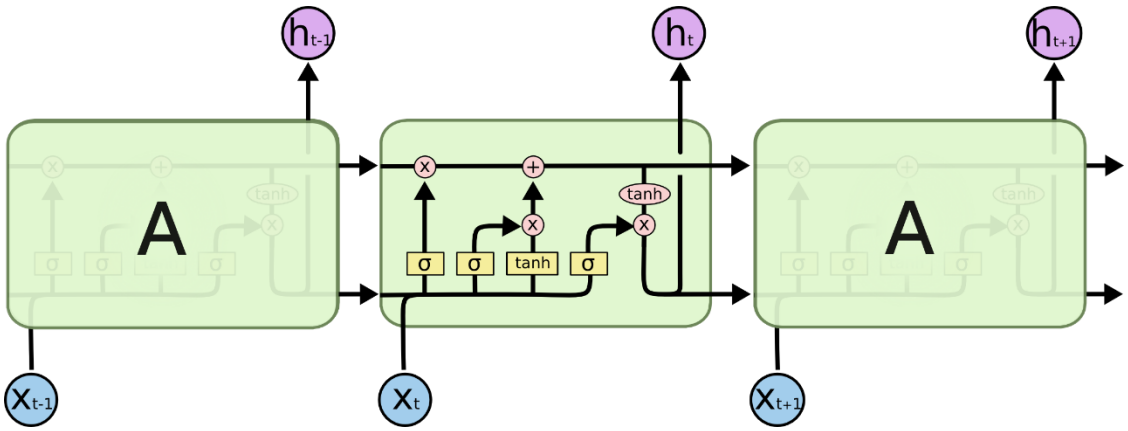
To čini LSTM idealnom NN arhitekturom za probleme poput sentimentalne analize i izvlačenja kompleksnih semantičkih značenja iz teksta, gdje se kontekst za neki pojam ili neku rečenicu može javiti na relativno „udaljenom“ mjestu u tekstu, npr. u nekoj prijašnjoj rečenici.

RNN mreže sastoje se od ulančanih ponavljajućih modula. U standardnim RNN mrežama takav modul ima vrlo jednostavnu strukturu, kao npr. jedan sloj s hiperboličkom tangens funkcijom (Slika 2.2).



Slika 2.2 Jednostavni ulančani RNN moduli [8]

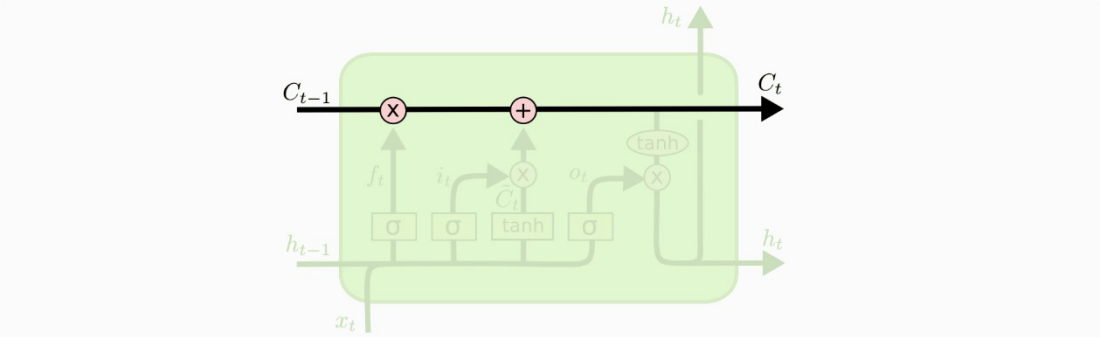
LSTM mreže također imaju lančanu strukturu, ali razliku od RNN mreža koje imaju jedan sloj, postoje 4 sloja koji međudjeluju na karakterističan način (Slika 2.3).



Slika 2.3 Ulančani četveroslojni LSTM moduli [9]

U gornjem dijagramu, svaka linija prenosi svojevrstan vektor, s izlaza jedne komponente na ulaz druge. Žuti pravokutnici zajedno predstavljaju 4 spomenuta sloja neuronske mreže, dok ružičasti krugovi predstavljaju operacije nad vektorima, kao zbrajanje i množenje vektora.

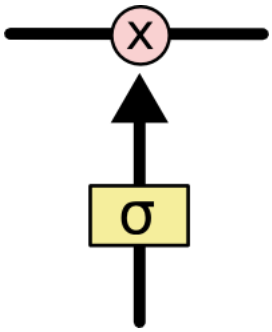
Ključna komponenta u LSTM je stanje ćelije (Slika 2.4), vidljiva na vrhu dijagrama. Stanje ćelije je kao pokretna traka, koja ide kroz sve module u lancu. Informaciji je vrlo lako teći uz ovu traku bez da se značajno mijenja, izuzev malih linearnih interakcija s modulima.



Slika 2.4 Stanje ćelije u LSTM modulu [10]

LSTM ima sposobnost uklanjanja i dodavanja informacije u stanje ćelije, što se pažljivo regulira strukturama zvanim vrata. Vratima se opcionalno propušta informacija prema stanju ćelije. Vrata se sastoje od sloja neuronske mreže sa sigmoidnom funkcijom te od operacije vektorskog množenja.

U nastavku je prikazana slika sloja sigmoidne funkcije (Slika 2.5):



Slika 2.5 Sloj sigmoidne funkcije (sigmoidni sloj) [11]

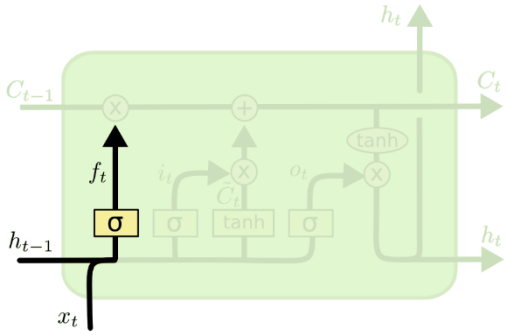
Sloj sigmoidne funkcije na izlazu generira vrijednost između 0 i 1, opisujući koliko informacije bi se trebalo propustiti. Vrijednost 0 označava nepropusnost dok vrijednost 1 označava propuštanje svega što dođe na ulaz.

LSTM ima tri takva vrata, kojima kontrolira i štiti stanje ćelije.

Prvi korak u LSTM je odlučivanje o tome koja će se informacija izbaciti iz stanja ćelije. Tu odluku donose vrata zaborava (engl. *forget gate*). Vrata zaborava u obzir uzimaju ulazne vrijednosti h_{t-1} i x_t, te na izlazu generiraju broj između 0 i 1 za svaki broj u stanju ćelije C_{t-1}, te se ovisno o tom broju informacija dijelom odbacuje ili zadržava.

LSTM je pogodan za problem analize sentimenta na način da pamti stvari te tako ima sposobnost izvlačenja konteksta iz svih prijašnje iznesenih riječi kako bi odredio značenje trenutne riječi. Na primjer, stanje ćelije može sadržavati kontekst negacije koji se primjenjuje na nadolazeće riječi, pamteći u stanju ćelije da se nadolazeće riječi trebaju promatrati u negiranom smislu. Konkretno, izjava „Nije da mrzim ovaj film“ treba se obraditi na način da se izvede točan sentiment, tako što se glagol „mrzim“ ne uzme kao riječ negativnog sentimenta, nego se u obzir uzme i prijašnja negacija te riječi pomoću fraze „Nije da“.

Vrata zaborava (Slika 2.6) služe da zaboravimo na stari subjekt kad uočimo novi subjekt. Npr. u jednoj rečenici se referiramo na sentiment o filmu, npr. „Ovaj film mi se dopao.“, dok se u drugoj rečenici referiramo na sentiment o drugom, novom subjektu, kao o nekom liku iz filma („Mrzim osobnost tog zlikovca.“), stoga je potrebno disocirati novi sentiment od starog subjekta (tako da se glagol „mrzim“ ne odnosi na film nego na zlikovca, što može biti pozitivan sentiment na način da označava autorovu hvalu upućenu uvjerljivosti zlikovca).



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

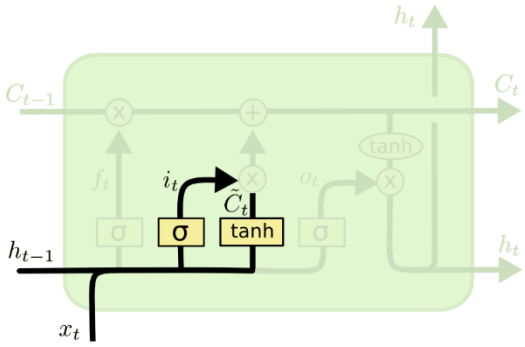
Slika 2.6 Vrata zaborava [12]

Sljedeći korak je odlučivanje o tome koja nova informacija će se spremiti u stanje ćelije.

Proces odlučivanja se sastoji od dva dijela. Prvo, sigmoidni sloj zvan „sloj vrata ulaza“ (engl. *input gate layer*) odlučuje koje će se vrijednosti izmijeniti (Slika 2.7). Potom, sloj hiperboličnog tangensa generira vektor kandidata za nove vrijednosti, $\sim C_t$, koje će potencijalno biti pridodane stanju ćelije.

U sljedećem koraku, ta dva dijela odluke kombiniraju se i time rade ažuriranje vrijednosti stanja. U primjeru sa sentimentom filma, želimo zaboraviti sentiment prema prošlom

subjektu (film) i dodati sentiment novog subjekta (zlikovca) u stanje ćelije, tako da zamjeni starog subjekta kojeg zaboravljamo.

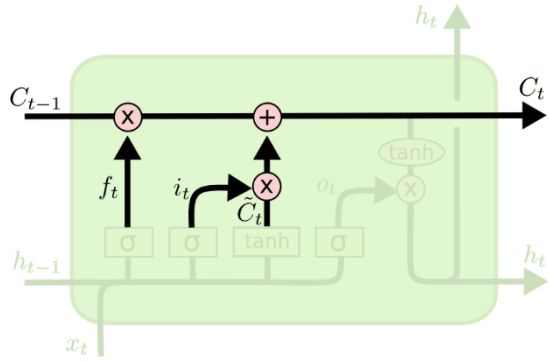


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Slika 2.7 Sloj vrata ulaza [13]

Sada na red dolazi ažuriranje starog stanja ćelije, C_{t-1}, u novo stanje ćelije C_t, na način na koji je odlučeno u prijašnjim koracima. Staro stanje množi se s f_t, čime se zaboravlja ono što se ranije i odlučilo da se zaboravlja. Onda se na to pridodaje $i_t * \sim C_t$, što predstavlja kandidate za nove vrijednosti, skalirane s faktorom koji određuje koliko puno se pojedina vrijednost u stanju želi ažurirati.

U ovom koraku se zapravo izvršava ono što se odlučilo u prijašnja dva koraka (zaboravljanje na starog subjekta i zamjena informacijom o novom subjektu) (Slika 2.8).

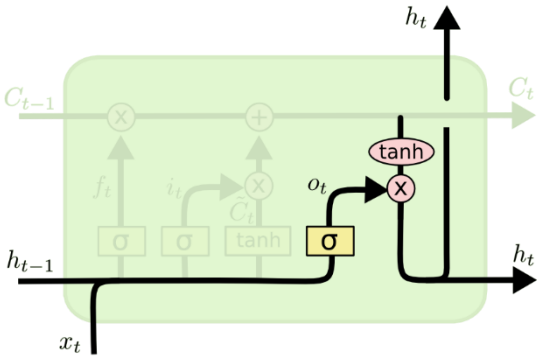


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Slika 2.8 Ažuriranje stanja ćelije [14]

Konačno, odlučuje se o tome što se šalje na izlaz modula. Izlaz ovisi o stanju ćelije, koje se dodatno filtrira. Prvo, sigmoidni sloj odlučuje koji dijelovi stanja ćelije će se slati na izlaz. Potom, nad stanjem ćelije primjenjuje se funkcija hiperboličnog tangensa, tako da se vrijednosti preslikaju na interval između -1 i 1. Potom se rezultat funkcije hiperboličnog tangensa množi izlazom iz sigmoidnih vrata tako da se na izlaz šalju samo dijelovi za koje je odlučeno da se šalju (Slika 2.9).

Na primjeru sentimenta o filmu, modul koji obrađuje subjekt u idealnom slučaju odabrat će relevantne informacije da bi ih proslijedio na izlaz za preciznije poštivanje konteksta u nadolazećim ćelijama [15].

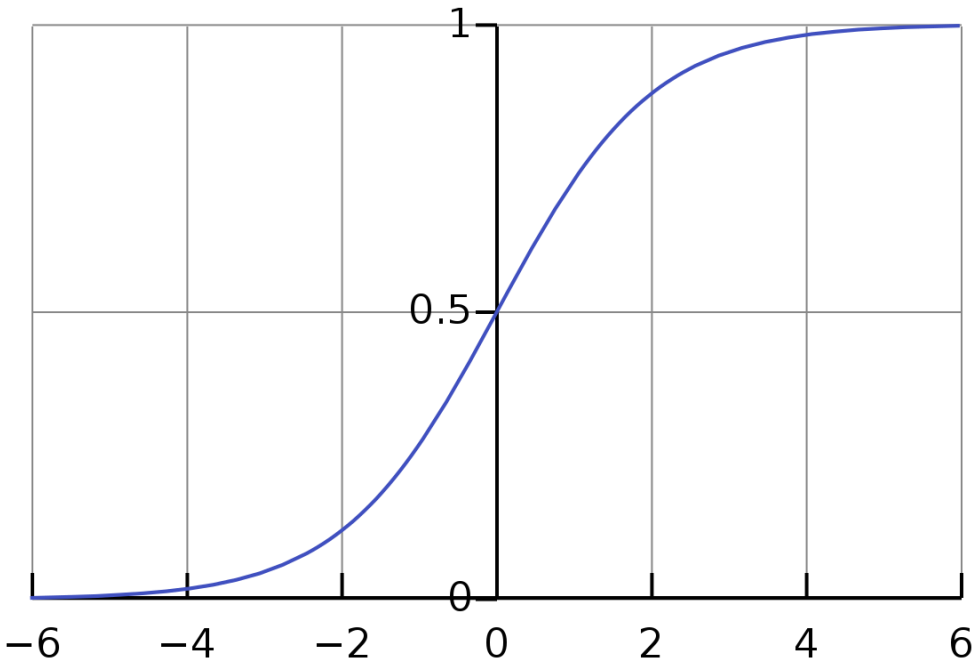


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Slika 2.9 Izlaz LSTM modula [16]

2.1.3. Gusto popunjeni (*Dense*) sloj

Posljednji, izlazni sloj u mreži je *Dense* sloj veličine 1, sa sigmoidnom aktivacijskom funkcijom (Slika 2.10), u svrhu binarnog klasificiranja.



Slika 2.10 Sigmoidna funkcija [17]

Takva funkcija na izlazu daje vrijednosti između 0 i 1, pa je prirodno koristiti takvu funkciju za binarnu klasifikaciju, npr. vrijednosti bliže 0 označavat će negativni sentiment a vrijednosti bliže 1 pozitivni sentiment.

2.2. Evolucijski algoritam

Evolucijski algoritam (EA) je metaheuristički optimizacijski algoritam inspiriran prirodom te se oslanja na koncepte iz darvinističke evolucije.

U EA, potencijalna rješenja optimizacijskog problema modeliraju se individualnim organizmima tj. jedinkama koje zajedno čine populaciju. Na početku je populacija ispunjena slučajnim rješenjima iliti jedinkama. Potom se testira učinkovitost jedinki u rješavanju zadanog problema, na način da se svakoj jedinki mjeri funkcija dobrote.

Dobre jedinke, one koje imaju visoku vrijednost funkcije dobrote, odabiru se za reprodukciju (s velikom vjerojatnošću), čime osiguravaju prijenos svojih (potencijalno dobrih) gena u sljedeću generaciju populacije. Zatim se populacija ponovno evaluira te lošije jedinke bivaju eliminirane (s velikom vjerojatnošću).

Po uzoru na prirodu, jedinke u populaciji pri EA oslanjaju se na funkcije kao što su selekcija, reprodukcija, križanje i mutacija. Prilagodni proces odabira najboljeg rješenja nekog problema gdje se javlja selekcija bazirana na funkciji dobrote / učinkovitosti – analogno je darvinističkom opstanku najjačih (engl. *survival of the fittest*).

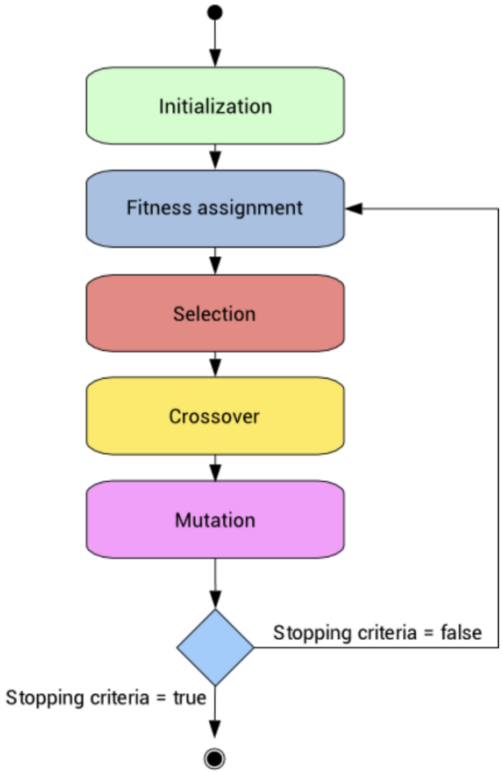
Mjerenjem funkcije dobrote kroz dobivene performanse pojedinih jedinki, kroz generacije se počinje javljati optimizacija, uz pomoć navedenih funkcija kao što su mutacija, selekcija…

EA je vrlo prilagodljiv i dobro aproksimira rješenja za razne tipove problema jer u principu nema pretpostavki o izgledu funkcije dobrote [18].

Bitno je naglasiti da iako se EA učinkovito bliži optimumu, ne mora uvijek naći optimalno rješenje, zbog načina na koji traži rješenje – a to je međusobnim uspoređivanjem funkcije dobrote jedinki, u neznanju o tome što bi zapravo bilo optimalno.

U ovom projektu jedinke se manifestiraju kao konfiguracije težinskih vrijednosti u neuronskoj mreži, to jest, jedinke smatramo analogne neuronskim mrežama, nad kojima vršimo spomenute funkcije evolucije, kao mutacija, selekcija… Takav pristup rješavanju optimizacijskih problema zovemo neuroevolucijom [19].

Na Slika 2.11 prikazane su faze općenitog evolucijskog algoritma:



Slika 2.11 Dijagram faza evolucijskog algoritma [20]

Kriterij zaustavljanja (engl. *stopping criteria*) je u okviru ovog rada implementiran kao opcionalni parametar koji označava redni broj posljednje generacije koja će se generirati.

2.2.1. Funkcija dobrote

Funkcija dobrote (engl. *fitness*) jedinke je mjera učinkovitosti jedinke u rješavanju zadanog optimizacijskog problema te je ujedno i odlučujući faktor u tome da li se geni neke jedinke propagiraju u sljedeću generaciju, i s kojom vjerojatnošću se to čini.

U okviru ovog projekta, funkcija dobrote jedinke jednaka je preciznosti (engl. *accuracy*) koja se dobije kao metrika prilikom evaluacije modela neuronske mreže (tj. jedinke) pomoću Keras API.

Metrika preciznosti gleda se kao točnost zaključivanja neuronske mreže nad priloženim skupom podataka, te u kojoj mjeri neuronska mreža za priložene recenzije filmova na ulazu daje točne klasifikacije po sentimentu na izlazu. Najbolja preciznost iznosi 1 (100% točnih predviđanja klase recenzije po sentimentu), dok najgora iznosi 0.5 (u 50% predviđanja

algoritam klasificira po jednom sentimentu, dok u drugih 50% predviđanja algoritam klasificira po drugom sentimentu, što je nalik na slučajan algoritam klasifikacije).

Trivijalan pseudokod za određivanje funkcije dobrote izložen je u nastavku:

```
def fja_dobrote(model):
     metrike_evaluacije = evaluiraj(model)
     vrati metrike_evaluacije[preciznost]
```

Daljnja obrada odgovarajuće jedinke bazira se na dobivenoj vrijednosti funkcije dobrote u odnosu na vrijednosti funkcije dobrote ostalih jedinki.

2.2.2. Selekcija

Faza selekcije je faza u kojoj se odabiru roditeljske jedinke za parenje i za generiranje jedinki koje će sačinjavati sljedeću generaciju. Ova faza je ključna za konvergenciju EA k dobrom rješenju jer dobre roditeljske jedinke općenito generiraju bolje potomke od ostalih jedinki.

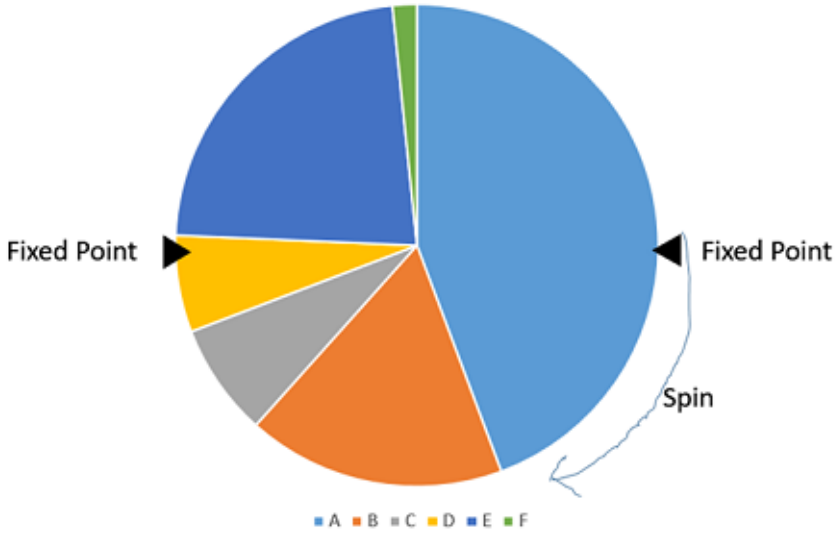
Međutim, također je potrebno spriječiti jedinku s vrlo visokom vrijednošću funkcijom dobrote da prevlada nad čitavom populacijom, jer to čini rješenja (koja su modelirana jedinkama) vrlo blizu jedno drugom u prostoru rješenja, pa se gubi raznolikost jedinki.

Održavanje određenog stupnja raznolikosti vrlo je važno za uspješnost algoritma, jer se inače dešava prijevremena konvergencija u neki lokalni optimum koji nije ujedno i globalni optimum.

Popularan način za odabir roditeljskih jedinki koje će generirati potomke za iduću generaciju je odabir proporcionalan funkciji dobrote (engl. *fitness proportionate selection*). Na ovaj način svaka jedinka ima vjerojatnost da će biti odabrana kao roditeljska jedinka proporcionalnu funkciji dobrote. Prirodnom selekcijom odabiru se jedinke veće vrijednosti funkcije dobrote i na taj način se propagiraju dobre kvalitete jedinki u sljedeću generaciju, pa kroz vrijeme jedinke evoluiraju i poboljšavaju se.

Veličina populacije jedinki je konstantna te se loše jedinke prirodnom selekcijom odbacuju, jer je mala vjerojatnost da će svoje značajke propagirati u sljedeću generaciju.

Na Slika 2.12 vizualiziran je jedan takav način selekcije gdje je vjerojatnost selekcije proporcionalna vrijednosti funkcije dobrote za odgovarajuću jedinku.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Slika 2.12 Stohastičko univerzalno uzorkovanje [21]

Stohastičko univerzalno uzorkovanje (engl. *stohastic universal sampling*) je kružno kolo koje modelira „kolo sreće“ po kojem se selektiraju dvije jedinke kao roditelji koji će generirati jednog potomka za sljedeću generaciju.

Svaka jedinka dobije odgovarajući kružni isječak s veličinom proporcionalnom funkciji dobrote te jedinke. Postoje dvije dijametralno suprotne fiksne točke koje će odrediti koje se jedinke odabiru kao roditelji. Kolo se „zavrti“ te se odabiru one jedinke na koje padnu fiksne točke. Stoga, vjerojatnost da će jedinka biti odabrana proporcionalna je njenoj vrijednosti funkcije dobrote.

Algoritam stohastičkog univerzalnog uzorkovanja opisan je pseudokodom [22] :

```
izračunaj S = sumu vrijednosti fje dobrote za sve jedinke

generiraj slučajni broj P iz [0,S>

počevši od najboljih jedinki, dodavaj vrijednosti fje dobrote
na parcijalnu sumu P, sve dok vrijedi P < S

jedinka za koju P premaši S se odabire, kao i njen
dijametralno suprotni par
```

2.2.3. Križanje

Križanje roditeljskih jedinki u evolucijskom algoritmu analogno je biološkom križanju, na način da se potomci generiraju pomoću genetskog materijala oba roditelja.

Jedan tip križanja za jedinke reprezentirane vektorima realnih brojeva je aritmetička rekombinacija (engl. *whole arithmetic recombination*). To je strategija križanja koja matematički miješa svaki gen oba roditelja, tako radeći novu djecu.

Aritmetička rekombinacija uzima određen udio gena jedne roditeljske jedinke i kombinira te gene s genima druge roditeljske jedinke. Fatkor α određuje u kojoj mjeri se uzima udio gena prvog ili drugog roditelja. Pri svakom križanju taj se faktor generira slučajno na intervalu [0,1].

Aritmetičkom rekombinacijom nastaju dva djeteta, čije formule glase:

- Dijete 1 = α * x + (1 - α) * y
- Dijete 2 = (1 - α) * x + α * y

gdje je x težinski vektor koji reprezentira prvog roditelja, a y težinski vektor koji reprezentira drugog roditelja [23].

2.2.4. Mutacija

Mutacija se može definirati kao mala slučajna izmjena u genomu, u svrhu dobivanja novog rješenja. Korisno je za unošenje raznolikosti u populaciju jedinki. Obično se primjenjuje s malom vjerojatnošću, jer ako se slučajne izmjene u genomu primjenjuju s prevelikom vjerojatnošću, EA biva reduciran na slučajno pretraživanje.

Mutacija je dio EA koji je povezan sa „pretraživanjem“ prostora rješenja zadanog optimizacijskog problema. Primijećeno je da je mutacija neophodna za konvergenciju EA, dok križanje nije [24].

Parametri jedinki za mutaciju odabiru se slučajno. Količina odabranih parametara ovisi o vjerojatnosti mutacije, npr. uz vjerojatnost mutacije $p_m = 0.02$ (tj. 2%), za mutaciju će biti odabran svaki pedeseti parametar, čime se mutira 2% parametara iz skupa svih parametara. U okviru ovog projekta, uz slučajan odabir parametara slučajna je i magnituda mutacije nad pojedinim parametrom, te ima normalnu razdiobu s pretpostavljenom standardnom devijacijom $\sigma = 0.1$.

2.2.5. Evaluacija

Evaluacijom određuje se preciznost jedinki nad skupom podataka za testiranje, to jest mjera točnosti klasifikacije recenzija po sentimentu, analogno funkciji dobrote koja se računala nad skupom podataka za učenje.

U Kerasu je radi toga ponovno korištena metrika *accuracy* koja označava mjeru točnosti klasifikacije nad skupom podataka, u intervalu [0,1].

O rezultatima evaluacije među ostalim ovise faza prilagodbe te odabir „elitnih“ jedinki.

2.2.6. Prilagodba

Rezultati evaluacije utječu na vjerojatnost mutacije i magnitudu mutacije, na način da se svakih 5 generacija uzme u obzir preciznost najbolje jedinke, te se ovisno o njoj vjerojatnost i magnituda mutacije povećaju ili smanje. Pri početnim generacijama populacije, ima smisla poticati visok stupanj raznolikosti, kako bi se prostor rješenja pregledavao u širinu, čime se sprečava prijevremena konvergencija u lokalni optimum i potiče istraživanje čitavog prostora rješenja kako bi se moglo naći što bolje rješenje.

Za kasnije generacije ima smisla sužavati područje pretrage rješenja individualnih jedinki jer nije poželjno imati velike skokove u prostoru rješenja kad je već prisutna visoka preciznost pojedinih jedinki. Umjesto toga, potrebno je raditi male preinake u genomu kako bi jedinka što bolje pretražila usku okolinu dobrog rješenja, u potrazi za još boljim rješenjem.

Kao funkcija faktora sužavanja okoline pretrage, odabrana je sljedeća, naizgled arbitrarna funkcija (Slika 2.13):

Slika 2.13 Funkcija faktora sužavanja prostora pretraživanja

Na slici je prikazana ovisnost faktora sužavanja (vertikalna os) u odnosu na maksimalnu preciznost dobivenu evaluacijom populacije (horizontalna os). Ovisnost je modelirana funkcijom:

$$f(x) = 1 / (10 * x + 1) - 0.09$$

Takva funkcija drastično mijenja nagib u odnosu na to je li preciznost mala ili velika, omogućavajući dvije različite metode operiranja (široka pretraga vs. uska pretraga).

Intuitivno, želi se vrlo široka pretraga tj. visoka razina mutacije ako je maksimalna preciznost jedinki relativno niska, tako da se potencijalna rješenja gledaju na drugim, udaljenim mjestima u prostoru koja potencijalno sadrže bolja rješenja.

S druge strane, želi se relativno uska pretraga ako je maksimalna preciznost visoka, tako da se detaljno pretraži usko okruženje dobrih rješenja u potrazi za još boljim rješenjima.

Ovakav oblik funkcije je također pogodan zbog sjecišta s osima na koordinatama koje su približne „ekstremima“, gdje su ekstremi preciznost od 0 (faktor približno 1 tj. 100%-ni učinak zadane mutacije) i preciznost od 1 (faktor približno 0 tj. učinak mutacije je blizu 0%

radi detaljne pretrage bez velikih skokova). Faktor sužavanja prostora množi se sa zadanim parametrima vjerojatnosti i magnitude mutacije, pri svakom generiranju nove populacije.

2.2.7. Elitizam

Kako bi se zadržao monotoni rast maksimalne preciznosti rješenja kroz generacije, upotrebljava se strategija zvana elitizam. Elitizam znači da se nekolicina najboljih jedinki trenutne generacije uvijek propagira u sljedeću generaciju, te one imaju garanciju da neće biti zamijenjene nekom drugom jedinkom prelaskom u sljedeću generaciju. Elitizam u obzir uzima mjeru preciznosti u fazi evaluiranja.

Kod elitizma važno je osigurati da se najbolje jedinke ne izbace iz populacije, ali isto tako je važno održavati raznolikost, pa se preporuča da veličina elitnog dijela populacije bude relativno malena. U okviru ovog projekta u obzir se uzima najboljih 10% jedinki kao elitne jedinke koje imaju zagarantirano mjesto u populaciji u sljedećoj generaciji [25].

Elitne jedinke se same po sebi ne mutiraju jer moraju ostati iste za sljedeću generaciju, ali one sudjeluju u križanju pa se njihova djeca mogu mutirati. Time i elitne jedinke sudjeluju u izradi genoma jedinki nove generacije.

3. Rezultati

U nastavku su rezultati optimizacije najznačajnijih parametara u radu algoritma neuroevolucije, a to su:

- veličina populacije
- vjerojatnost mutacije
- magnituda mutacije

Optimiranje se provodi odabirom vrijednosti parametara od kojih svaka vrijednost reprezentira određeni red veličine (npr. red veličine 0.01, 0.1, 1…), te nalaženjem vrijednosti koja daje maksimalnu preciznost tj. maksimalnu vrijednost funkcije dobrote.

Za optimizaciju svakog od navedenih parametara algoritam je izvršen 20 puta do kriterija zaustavljanja kako bi se ustanovio prosjek rješenja, što je potrebno raditi zbog stohastičke naravi algoritma radi objektivne analize.

No, prije optimiranja parametara potrebno je ustanoviti kriterij zaustavljanja.

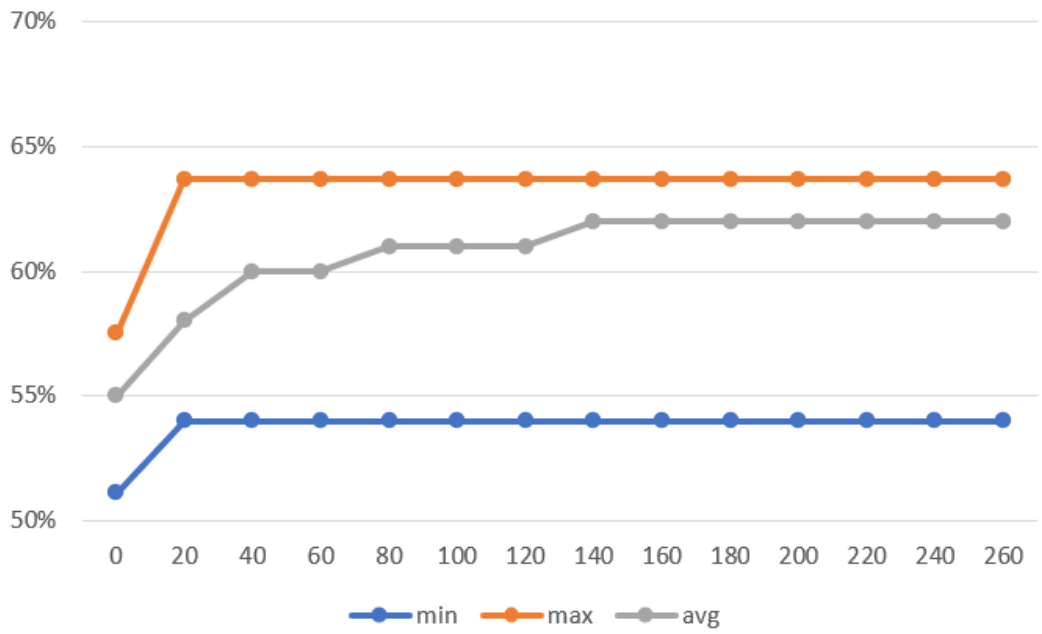
3.1. Kriterij zaustavljanja

U kontekstu ovog rada, kriterij zaustavljanja se definira kao redni broj generacije gdje daljnje izvođenje algoritma više nema značajan učinak na preciznost.

U svrhu traženja kriterija zaustavljanja, algoritam je izvršen 20 puta, gdje su parametri algoritma postavljeni na sljedeće arbitrarne vrijednosti:

- veličina populacije = 50
- vjerojatnost mutacije = 0.01
- magnituda mutacije = 0.1

Na Slika 3.1 vidljivo je kako algoritam nema smisla izvoditi nakon 160 generacija jer algoritam nakon tog broja generacija više ne napreduje:



Slika 3.1 Dijagram preciznosti u ovisnosti o broju generacija

Vidljivo je kako algoritam u prosjeku monotono povećava preciznost te staje na približno 62% prosječne preciznosti, te se nakon 160.-te generacije rješenje ne poboljšava.

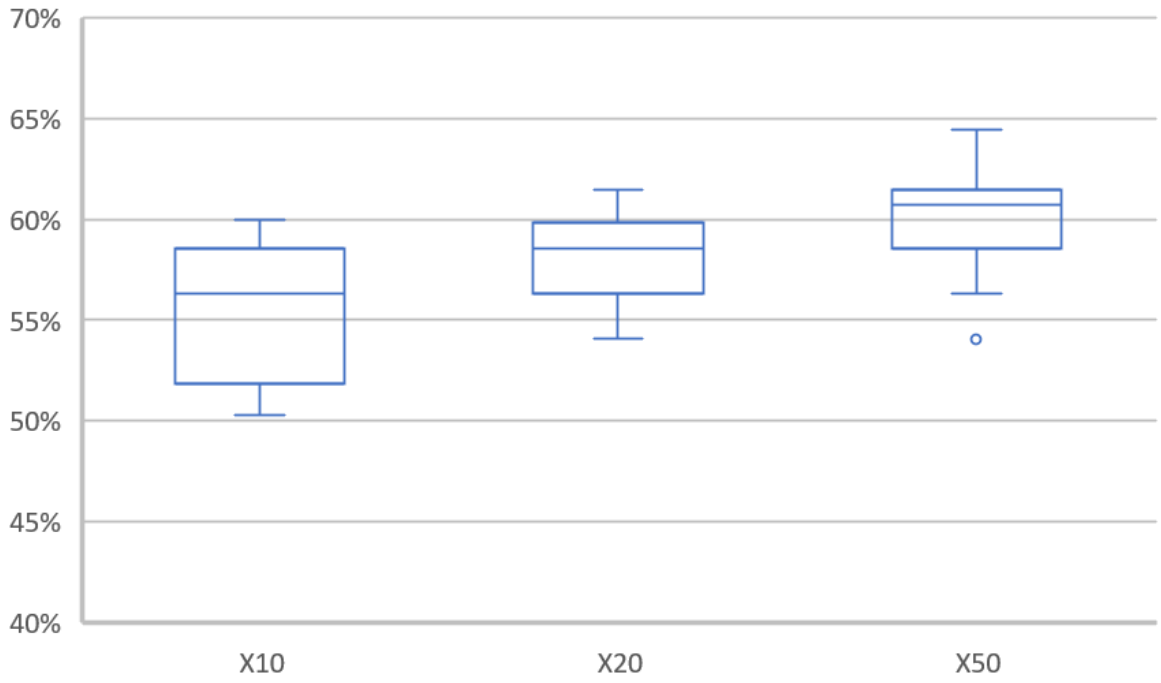
Maksimum je ravna linija nakon 20.-te generacije jer je unutar prvih 20 generacija u nekom eksperimentu nađeno najbolje rješenje približne preciznosti 64.5%.

Minimum je ravna linija nakon 20.-te generacije jer u nekim eksperimentima napredovanje iznad 54.5% preciznosti ne postoji nakon 20.-te generacije. Razlog tome mogu biti loši početni geni, u kombinaciji s relativno malom populacijom, tako da je mala vjerojatnost da će ijedna jedinka „naletjeti“ na bolje rješenje (mutacijom, križanjem…).

3.2. Optimizacija veličine populacije

Isprobane su sljedeće veličine populacije: 10, 20, 50.

Na Slika 3.2 prikazan je odnos maksimalne preciznosti (koja se postigla u seriji od 20 eksperimenata za svaku spomenutu vrijednost veličine populacije) i veličine populacije.



Slika 3.2 Dijagram ovisnosti maksimalne preciznosti o veličini populacije

S dijagrama je vidljivo da se za veličinu populacije 50 dobiva najveća preciznost, kako u prosjeku tako i po maksimalnoj preciznosti. Ta maksimalna preciznost iznosi 64.44%.

Hipotetski, preciznost raste proporcionalno veličini populacije, iz razloga što tada postoji više jedinki koje pretražuju prostor definiran funkcijom dobrote, pa je veća vjerojatnost da će neka jedinka naći dobro rješenje te svoje dobre gene propagirati u sljedeću generaciju.

Kod populacija s manjim brojem jedinki, postoji manji skup gena iz kojeg se mogu izvlačiti nova rješenja, pa je veća vjerojatnost da će sva rješenja biti međusobno slična. Takav problem „recikliranja“ gena kroz generacije izražen je u manjim populacijama, a ne toliko u velikim populacijama, jer velike populacije po definiciji imaju mnogo „različitih“ rješenja tj. relativno visok stupanj raznolikosti (engl. *diversity*), te križanjem takvih rješenja ne uvodi se pretjerana međusobna sličnost rješenja na globalnoj razini populacije. Tome doprinosi i činjenica da se nad svakim generiranim rješenjem provodi mutacija, pa je u većim populacijama manja vjerojatnost da će postojati problem „recikliranja“ tj. problem kontinuirane sličnosti rješenja kroz generacije.

Stoga, pretpostavlja se da veće populacije mogu dosegnuti veću maksimalnu preciznost od manjih populacija za isti broj evaluacija jedinki.

No, za veće veličine populacije (npr. 100, 1000…) potrebna je velika hardverska moć i dulje vrijeme učenja, što pri izradi ovog projekta nije bilo dostupno. Svaka jedinka sadrži preko

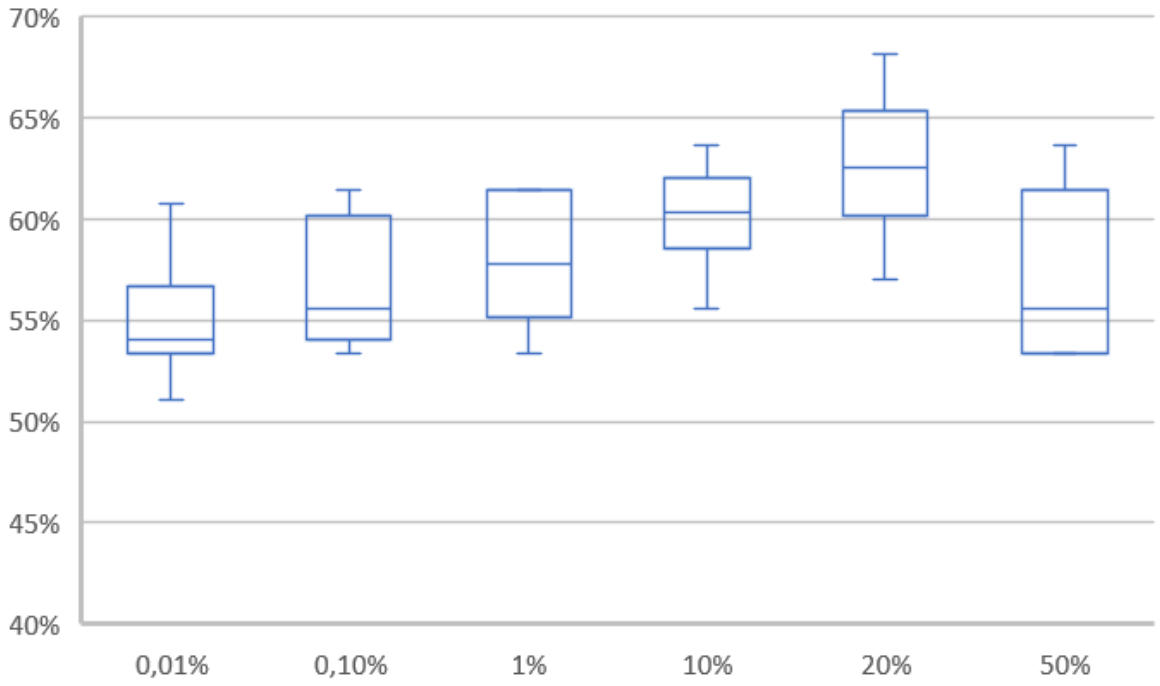
200 000 parametara te se svaka jedinka uči i evaluira prilikom svake nove generacije, što je hardverski zahtjevno za veće veličine populacije.

3.3. Optimizacija vjerojatnosti mutacije

Veličina populacije postavljena je na vrijednost 50 za koju je zaključeno da je optimalna (nad skupom 10, 20 i 50).

Isprobane su sljedeće vjerojatnosti mutacije: 0.01%, 0.1%, 1%, 10%, 20%, 50%.

Na Slika 3.3 prikazan je odnos maksimalne preciznosti i vjerojatnosti mutacije.



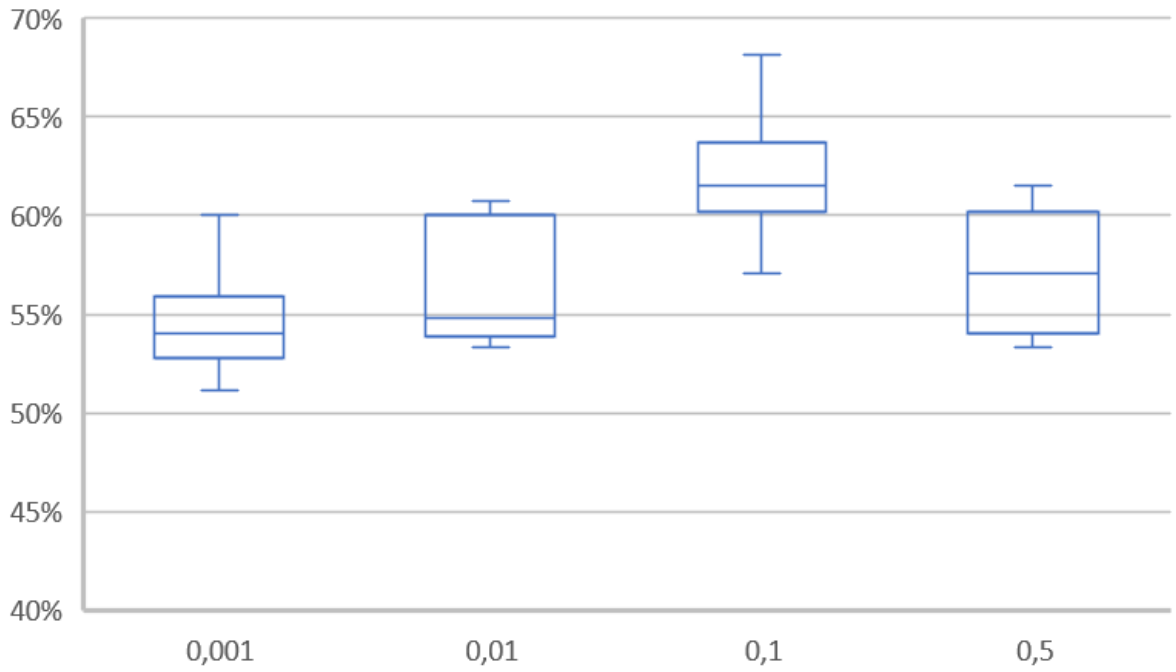
Slika 3.3 Dijagram ovisnosti maksimalne preciznosti o vjerojatnosti mutacije

Na dijagramu je vidljivo da se maksimalna preciznost postiže s vjerojatnošću mutacije postavljenom na 20%. Maksimalna preciznost za tu vjerojatnost mutacije je iznosila 68.15%, a u prosjeku preciznost je bila 62.52%.

3.4. Optimizacija magnitude mutacije

Isprobane su magnitude mutacije: 0.001, 0.01, 0.1, 0.5.

Na Slika 3.4 prikazan je odnos maksimalne preciznosti i magnitude mutacije.



Slika 3.4 Dijagram ovisnosti maksimalne preciznosti o magnitudi mutacije

S dijagrama je vidljivo da je optimalna magnituda mutacije reda 0.1, za koju se dobiva maksimalna preciznost 68.15% i prosječna preciznost 61.70%.

Magnituda mutacije 0.5 nije idealna jer tada jedinke mutacijom rade prevelike skokove u prostoru funkcije dobrote da bi detaljno istraživale prostor za dobro rješenje.

S druge strane magnituda mutacije 0.001 čini da jedinke rade premale skokove i ne pretražuju prostor dovoljno u širinu, čime se zadržavaju na lokalnim optimumima.

3.5. Mjerodavni rezultati

Mjerodavna *state-of-the-art* maksimalna preciznost (tj. *benchmark*) u analizi sentimenta nalazi se u rangu od 70.5% za jednostavne modele do 81.5% za nešto složenije modele [26], što se postiže klasičnim metodama analize sentimenta poput naivnog Bayesovog klasifikatora, dubokog učenja pomoću LSTM, dubokog učenja s unaprijed istreniranim *Embedding* slojem… [27]

Maksimalna postignuta preciznost pomoću neuroevolucije u ovom projektu iznosi 68.15%, što nije previše ispod ranga konvencionalnijih metoda analize sentimenta.

Zaključak

Analiza sentimenta je složen problem kojeg nije jednostavno riješiti na način da se postigne veliki stupanj preciznosti u raspoznavanju sentimenta. Potrebna je velika hardverska moć i puno vremena za učenje i optimiranje algoritma da bi se dobili značajni rezultati.

U okviru ovog rada, algoritmom neuroevolucije u kombinaciji s LSTM neuronskim mrežama, problemu sentimentalne analize pristupilo se iz novog kuta. Rezultati su očekivano ispod rezultata konvencionalnih *state-of-the-art* algoritama.

No, svejedno se nazire potencijal neuroevolucije kao metaheuristički pristup analizi sentimenta, s maksimalnom preciznošću 68.15% postignutom u ovom radu, što je relativno zadovoljavajuće u usporedbi s mjerodavnom donjom granicom preciznosti konvencionalnih algoritama 70.5%.

Veća preciznost mogla bi se postići učenjem populacija veće veličine, temeljitijom optimizacijom parametara, kombiniranjem neuroevolucije s prenesenim znanjem (engl. *transfer* learning) tako da se koristi unaprijed naučeni *Embedding* sloj, i slično.

Literatura

[1] *Sentiment Analysis: A Definitive Guide*, Monkey Learn. https://monkeylearn.com/sentiment-analysis/; pristupljeno 3.6.2022.

[2] *Large Movie Review Dataset*, https://ai.stanford.edu/~amaas/data/sentiment/; pristupljeno 3.6.2022.

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts, Learning Word Vectors for Sentiment Analysis, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (2011.).

[4] *The difference between Euclidean distance and cosine similarity*. https://www.researchgate.net/publication/320914786/figure/fig2/AS:558221849841664@1510101868614/The-difference-between-Euclidean-distance-and-cosine-similarity.png; pristupljeno 3.6.2022.

[5] Francois C., *Using pre-trained word embeddings in a Keras model*, The Keras Blog, (2016, Srpanj), https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html; pristupljeno 3.6.2022.

[6] Jason B., *How to Use Word Embedding Layers for Deep Learning with Keras*, Machine Learning Mastery, (2017, listopad). https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/; pristupljeno 3.6.2022.

[7] *Long short-term memory*, Wikipedia, (2022, svibanj). https://en.wikipedia.org/wiki/Long_short-term_memory; pristupljeno 3.6.2022.

[8] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-SimpleRNN.png; pristupljeno 10.6.2022.

[9] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain.png; pristupljeno 10.6.2022.

[10] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-C-line.png; pristupljeno 10.6.2022.

[11] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-gate.png; pristupljeno 10.6.2022.

[12] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-f.png; pristupljeno 10.6.2022.

[13] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-i.png; pristupljeno 10.6.2022.

[14] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-C.png; pristupljeno 10.6.2022.

[15] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/img/LSTM3-focus-o.png; pristupljeno 10.6.2022.

[16] *Understanding LSTM Networks*, colah's blog, (2015, kolovoz). http://colah.github.io/posts/2015-08-Understanding-LSTMs/; pristupljeno 3.6.2022.

[17] Wikipedia. https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/1200px-Logistic-curve.svg.png; pristupljeno 10.6.2022.

[18] *Evolutionary algorithm*, Wikipedia, (2022, svibanj). https://en.wikipedia.org/wiki/Evolutionary_algorithm; pristupljeno 3.6.2022.

[19] TechTarget Contributor, *Evolutionary algorithm*, (2018, travanj). https://www.techtarget.com/whatis/definition/evolutionary-algorithm; pristupljeno 3.6.2022.

[20] Medium. https://miro.medium.com/max/1400/1*uI6OYFgWwG_ngihySpybQA.png; pristupljeno 3.6.2022.

[21] *Genetic Algorithms – Parent Selection*, tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/images/sus.jpg; pristupljeno 10.6.2022.

[22] *Genetic Algorithms – Parent Selection*, tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm; pristupljeno 3.6.2022.

[23] *Genetic Algorithms – Crossover*, tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_crossover.htm; pristupljeno 3.6.2022.

[24] *Genetic Algorithms – Mutation*, tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_mutation.htm; pristupljeno 3.6.2022.

[25] *Genetic Algorithms - Survivor Selection*, tutorialspoint. https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_survivor_selection.htm: pristupljeno 3.6.2022.

[26] *Sentiment Accuracy: Explaining the Baseline and How to Test It*, Lexalytics. https://www.lexalytics.com/blog/sentiment-accuracy-baseline-testing/; pristupljeno 10.6.2022.

[27] Harleen Kaur, Shafqat Ul Ahsaan, Bhavya Alankar & Victor Chang, *A Proposed Sentiment Analysis Deep Learning Algorithm for Analyzing COVID-19 Tweets*, Springer Link (2021, travanj). https://link.springer.com/article/10.1007/s10796-021-10135-7; pristupljeno 10.6.2022.

Sažetak

Analiza sentimenta u recenzijama filmova pomoću neuroevolucije

Korišten je IMDB skup podataka za analizu sentimenta. Izgrađena je LSTM neuronska mreža pomoću koje se vrši evaluacija pojedinih potencijalnih rješenja. Evolucijski algoritam uzdržava i evoluira populaciju potencijalnih rješenja. Rješenja se evoluiraju pomoću prirodom inspiriranih procesa kao što su selekcija, mutacija, križanje i fitness tj. funkcija dobrote jedinki u populaciji.

Ključne riječi: neuronske mreže (NN), LSTM, analiza sentimenta, evolucija, evolucijski algoritam (EA), selekcija, mutacija, križanje, funkcija dobrote (fitness), elitizam, RNN, optimizacija

Summary

Sentiment analysis applied to movie reviews using neuroevolution

IMDB movie review dataset is used for sentiment analysis. LSTM neural network is built, and serves to evaluate individual potential solutions. Evolutionary algorithm is tasked with maintaining and evolving a population of potential solutions. Solutions are evaluated through processes inspired by nature, like selection, mutation, crossover and fitness of units within the population.

Keywords: neural networks (NN), LSTM, sentiment analysis, evolution, evolutionary algorithm (EA), selection, mutation, crossover, fitness, elitism, RNN, optimisation

Skraćenice

LSTM	*Long Short-Term Memory*	duga kratkoročna memorija
NLP	*Natural Language Processing*	obrada prirodnog jezika
NN	*Neural Network*	neuronska mreža
RNN	*Recurrent Neural Network*	povratna neuronska mreža
EA	*Evolutionary Algorithm*	evolucijski algoritam