Razvoj LSTM neuronske mreže i primena nad problemom sekvencijalnog učenja

Seminarski rad u okviru kursa
Računarska inteligencija
Matematički fakultet

Milena Kurtić, Nevena Soldat
mimikurtic67@gmail.com, nevenasoldat@gmail.com

17. april 2020.

Sažetak

U okviru ovog rada predstavljen je primer implementacije LSTM neuronske mreže u programskom jeziku Python, uz korišćenje biblioteka za rad kao što su pandas, numpy, keras i druge. Prvi deo rada opisuje teorijsku osnovu iza ovih mreža. Zatim je opisana primena na problem generisanja teksta. Videćemo koje su njene prednosti, kao i ograničenja. Pokazana je razlika u ponašanju modela sa različitim konfiguracijama.

Sadržaj

1 Uvod

Neuronske mreže (eng. neural networks) predstavljaju najpopularniju i jednu od najprimenjenijih metoda mašinskog učenja. Njihove primene su mnogobrojne i pomeraju domete veštačke inteligencije, računarstva i primenjene matematike. Postoji više vrsta neuronskih mreža: potpuno povezane, konvolutivne, rekurentne, grafovske neuronske mreže. U vrstu rekurentnih spada i LSTM neuronska mreza.

Osnovna ideja veštačke neuronske mreže je simulacija velike količine gusto napakovanih, međusobno povezanih nervnih ćelija u okviru računara, tako da je omogućeno učenje pojmova, prepoznavanje šablona i donošenje odluka na način koji je sličan čovekovom. Suštinski, veštačke neuronske mreže su softverske simulacije, napravljene programirajući obične računare koji rade u uobičajenom režimu sa svojim tranzistorima i serijski povezanim logičkim kolima, tako da se ponašaju kao da su napravljene od milijardu međusobno povezanih ćelija mozga koje rade paralelno.

1.1 Opis rada

U ovom radu je implementirana rekurentna neuronska mreža koja automatski generiše određeni tekst. Podaci koje koristimo predstavljaju kratke opise filmova. Cilj nam je da za određenu sekvencu reči uspemo da pretpostavimo koja će biti sledeća. Korišćena je posebna verzija rekurentne mreže koja ume da pamti duge sekvence - kratka dugoročna memorija (eng. Long Short Term Memory - LSTM).

Mnogi radovi bave se automatskim generisanjem teksta. Rad koji je sličan našem opisuje automatsko generisanje kratkih priča na nivou generisanja pojedinačnih reči [**?**]. Automatsko generisanje teksta može biti implementirano i na nivou karaktera, što je opisano u radu [**?**].

LSTM mreže koriste se za razne probleme nad sekvencijalnim podacima. Tako imamo radove koji se bave automatskim generisanjem reči pesama [**?**], ili čak automatskim komponovanjem muzike [**?**].

2 Rekurentne neuronske mreže

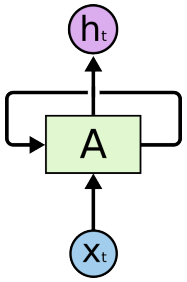
Rekurentne neuronske mreže (eng. Recurrent Neural Networks - RNN) predstavljaju arhitekturu mreža specijalizovanu za obradu sekvencijalnih podataka, poput rečenica prirodnog jezika i vremenskih serija. Sekvence nameću važnost redosleda zapažanja podataka, kako prilikom treniranja modela, tako i prilikom predviđanja. Mreže su konstruisane sa idejom da se modeluje zavisnost među instancama. Elementi ulazne sekvence se obrađuju u koracima, mreža ima skriveno stanje koje akumulira informaciju o elementima sekvence obrađenim u prethodnim koracima, a parametri određuju na koji način se to stanje menja iz koraka u korak na osnovu prethodnog stanja i tekućih ulaza i kako se generiše izlaz u zavisnosti od tekućeg stanja.

"Learning of sequential data continues to be a fundamental task and a challenge in pattern recognition and machine learning. Applications involving sequential data may require prediction of new events, generation of new sequences, or decision making such as classification of sequences or sub-sequences."

— On Prediction Using Variable Order Markov Models, 2004.

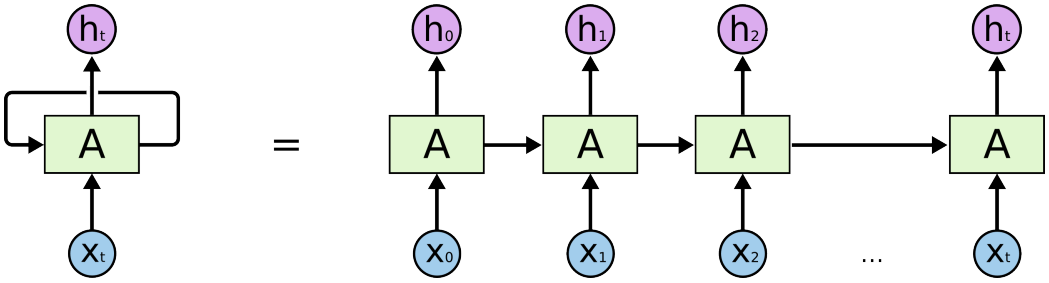
2.1 Arhitektura RNN

Rekurentne neuronske mreže sadrže petlje koje obezbeđuju čuvanje informacija [?].



Slika 1: Prikaz RNN mreže

Na slici 1 prikazan je jedan deo neuronske mreže, A, čiji je ulaz x_t, a izlaz h_t. Petlja omogućava da se informacije prosleđuju iz jednog koraka u mreži u drugi. Biće jednostavnije ako ih zamislimo na sledeći način:



Slika 2: Prikaz RNN mreže u obliku lanca

Rekurentna mreža predstavlja veći broj istih mreža, koje prenose poruke narednom delu u lancu, što je prikazano na slici 2.

2.2 Problemi sa RNN

Postoje dva osnovna problema rekurentnih neuronskih mreža u njihovoj osnovnoj formi. Prvi se tiče problema nestajućih i eksplodirajućih gradijenata. Naime, graf izračunavanja rekurentne neuronske mreže je tipično vrlo dubok zbog velike dužine sekvence. Usled toga, prilikom izračunavanja gradijenta propagacijom u prošlost, dolazi do velikog broja množenja koja neretko čine da koordinate gradijenta ili eksplodiraju ili nestanu.

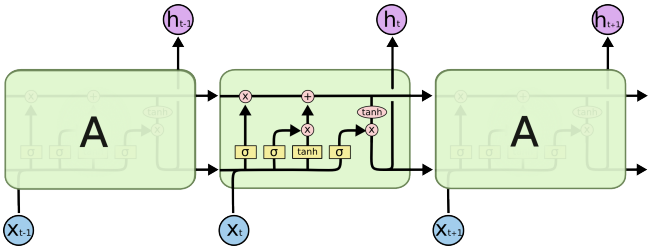
Drugi problem se odnosi na dugoročno čuvanje informacije i modelovanje dugoročnih zavisnosti u podacima. Kako se skriveno stanje u svakom koraku dobija linearnom kombinacijom prethodnog stanja i ulaza, doprinos starijih ulaza se brzo gubi pod uticajem novih. Dugoročno čuvanje relevantnih informacija nije moguće.

Oba ova problema se prevazilaze upotrebom duge kratkoročne memorije (eng. long short term memory), skraćeno LSTM, što je složena jedinica mreže sa specifičnom strukturom koja omogućava kontrolu čitanja i upisa u jedinicu. Upravo ova jedinica dovela je do ključnih uspeha rekurentnih

neuronskih mreža i predstavlja standardni izbor prilikom formulisanja modela rekurentne mreže. [?]

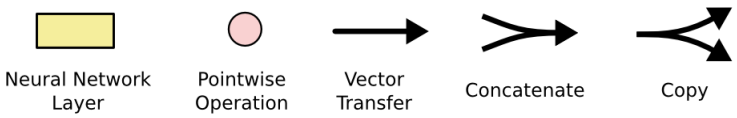
2.3 LSTM neuronske mreže

LSTM neuronske mreže su specijalizovane za prevazilaženje problema kratkotrajne memorije. Pamćenje dugih sekvenci je praktično njihovo podrazumevano ponašanje, a ne nešto sa čime se muče. Kao što je već prikazano, sve rekurentne mreže imaju lančanu formu, sastavljenu od modula mreže koji se ponavljaju. Kod LSTM mreža, struktura ovih modula je malo drugačija. Umesto jednog sloja mreže, imamo četiri koja su povezana na veoma poseban način (Slika 3).



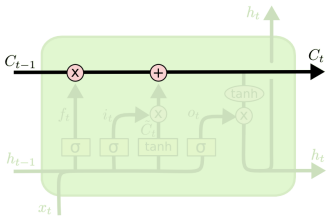
Slika 3: Prikaz lanca LSTM mreže

Na slici iznad, svaka linija prenosi vektor podataka, od izlaza iz prethodnog čvora, do ulaza u naredni. Roze krugovi predstavljaju operacije, poput sabiranja vektora, dok žuti pravougaonici predstavljaju slojeve mreže. Linije koje se spajaju obeležavaju spajanje informacija, dok linije koje se razdvajaju predstavljaju sadržaj koji se kopira, gde kopije idu na različita mesta (Slika 4).



Slika 4: Notacija

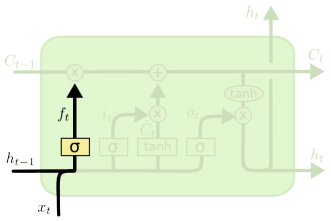
Osnovna ideja LSTM-a je postojanje takozvane ćelije koja čuva skriveno stanje, uz kontrolu pisanja, čitanja i zaboravljanja. Na slici 5 je to horizontalna linija na vrhu dijagrama.



Slika 5: Ćelija LSTM mreže

Stanje ćelije prolazi kroz ceo lanac, uz male linearne interakcije. Informacije se veoma lako prenose nepromenjene. LSTM može da doda ili ukloni informacije iz stanja ćelije, pomoću struktura koje se nazivaju kapije. Kapije odlučuju koje informacije i u kojoj količini će biti očuvane.

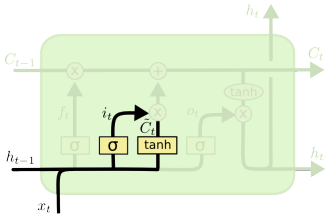
Prvi korak u našoj LSTM mreži je odabir informacija koje će biti obrisane iz stanja ćelije. Kapija koja kontroliše ovu operaciju naziva se „kapija zaboravljanja“. Ovaj sloj koristi sigmoidnu aktivacionu funkciju, čiji je izlaz interval brojeva između 0 i 1. Vrednost 1 znači da se potpuno zadržava sadržaj, a 0 da se potpuno briše (Slika 6).



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

Slika 6: Kapija zaboravljanja

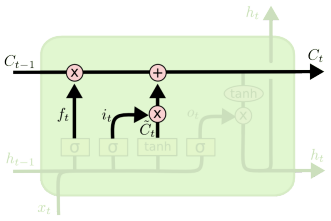
Sledeći korak predstavlja odlučivanje o tome koje nove informacije ćemo dodati u stanje ćelije. Sastoji se iz dva dela. Prvo, sloj koji predstavlja „ulaznu kapiju“ odlučuje koje će vrednosti ažurirati. Zatim naredni sloj, čija je aktivaciona funkcija tangens hiperbolički, pravi vektor novih vrednosti ($\tilde{C}_t$), koje se mogu dodati u stanje ćelije. U sledećem koraku kombinujemo ova dva dela i pravimo novo stanje ćelije (Slika 7).



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

Slika 7: Ulazna kapija

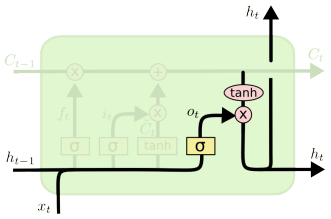
Sada prethodno stanje ćelije C_{t-1} ažuriramo u novo C_t. Pomnožimo staro stanje sa f_t, time zaboravljajući ono što smo odlučili da zaboravimo. Zatim dodamo $i_t \times \tilde{C}_t$. Ovo su potencijalne nove vrednosti, skalirane za onoliko koliko smo odlučili da ažuriramo svaku vrednost stanja (Slika 8).



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

Slika 8: Računanje novog stanja ćelije

Na kraju je potrebno da odlučimo šta će biti izlaz. Ova vrednost zavisi od stanja ćelije, uz male promene. Prvo prolazimo kroz sloj sa sigmoidnom aktivacionom funkcijom koji odlučuje koje delove ćelije ćemo staviti u izlaz. Zatim prođemo kroz aktivacionu funkciju tangens hiperbolički (kako bismo uokvirili vrednosti u interval između -1 i 1), i pomnožimo sa izlazom iz sloja sa sigmoidnom funkcijom kako bismo u izlaz stavili samo ono što smo odlučili (Slika 9).



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

Slika 9: Računanje izlaza

3 Problem generisanja teksta

Generisanje prirodnog jezika (eng. Natural Language Generation - NLG) se intenzivno proučava zahvaljujući bitnim primenama koje ima, kao što su automatsko generisanje dijaloga, automatsko prevođenje jezika, sumarizacija teksta, generisanje opisa fotografija, i mnogih drugih. Poboljšanje kvaliteta automatskog generisanja teksta (u smislu sintakse i semantike) dolazi sa primenom tehnika mašinskog učenja [**?**]. .

Ovaj rad bavi se automatskim generisanjem teksta na nivou reči. Podaci koje koristimo sadrže kratke opise filmova, preuzete sa vikipedije (mogu se preuzeti sa adrese: `https://www.kaggle.com/jrobischon/wikipedia-movie-plots`). Cilj je da se dobije kratak opis neke radnje filma, sa zadatim ulaznim tekstom. Ne očekujemo da rezultati budu precizni, bitno je da predviđeni tekst bude čitljiv.

Generisanje teksta vrši se tako da za datu početnu sekvencu teksta mreža predviđa sledeću reč, a zatim na osnovu novodobijene reči ponovo predviđa novu reč sve dok ne izgeneriše traženi broj reči. Faze rešavanja problema su sledeće:

- Preprocesiranje ulaznih podataka
- Kreiranje i treniranje modela
- Rezultati

3.1 Preprocesiranje ulaznih podataka

Ulazni podaci su preuzeti iz baze podataka koja sadrži radnje filmova. Na slučajan način biramo 100 filmova. Različiti filmovi imaju različite i radnje, pa je potrebno da model prvo razume reči i kontekst svake radnje.

Prvi korak u svim projektima koji podrazumevaju korišćenje reči prirodnog jezika je uklanjanje stop [1] reči. Ipak, nama je cilj generisanje teksta koji će ličiti na povezanu priču, te nam je potrebno da sačuvamo sve reči. Naredni koraci u obradi teksta podrazumevaju:

[1] ovo su uglavnom veznici, kao i reči koje su veoma učestale i koje iz tog razloga ometaju istraživanje

- konvertovanje svih slova u male
- uklanjanje znakova interpunkcije, kao i ostalih netipičnih karaktera
- uklanjanje belina
- ...

Dodatno, neuronske mreže ne mogu da rade sa rečima, te tekstualne podatke kodiramo kao brojeve koji će predstavljati ulaz, odnosno izlaz iz mreže. Tokenizer API iz biblioteke keras obavlja ovaj posao za nas. Naime, pozivom funkcije fit_on_texts obrađuje se tekst na gorenaveden način, i kreira se rečnik čiji su ključevi reči, a vrednosti brojevi koji su im dodeljeni.

3.1.1 Način predstavljanja teksta

Najjednostavniji način predstavljanja reči u metodama mašinskog učenja je vektor sa jednom jedinicom, tzv. 1-hot vektor. Ovim pristupom dobijaju se vektori velike dužine, a to u realnim primenama gde je vokabular koji se koristiri prilično velikih razmera nije optimalno. Drugi problem ovog pristupa je što algoritam nema nikakvo znanje o eventualnoj sličnosti između reči.

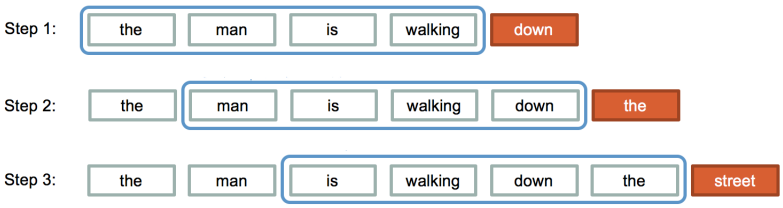
Pristup koji koristimo podrazumeva reprezentaciju reči u latentnom prostoru (eng. word embedding). Ovim pristupom vrši se redukcija dimenzionalnosti 1-hot reprezentacije u vektorski prostor čije koordinate nemaju direktno jasnog smisla čoveku (stoga i naziv latentni vektorski prostor). Dimenzionalnost tog vektorskog prostora zavisi od dalje primene, ali kreće se od nekoliko desetina do nekoliko hiljada. Ključna razlika u odnosu na 1-hot reprezentaciju je što je ona gusta (eng. dense), dok je prethodna retka (eng. sparse). Retke reprezentacije imaju svojih prednosti kada je u pitanju vremenska složenost nekih algoritama, prvenstveno onih u kojima ima operacija sa retkim vektorima i matricama. S obzirom na prirodu i arhitekturu neuronskih mreža, ova prednost ne postoji - neuronskim mrežama pogodna je gusta predstava, odnosno gusti vektori osobina reči.

Biblioteka Keras nam nudi Embedding sloj koji se može koristiti u ove svrhe. Sloj se inicijalizuje nasumičnim težinama, a zatim uči vektore svih reči pri treniranju skupa podataka. Sastoji se od tri argumenta:

- input_dim - veličina rečnika koji se koristi
- output_dim - dimenzija vektorskog prostora u kom su reči reprezentovane. Definiše dužinu vektora koji predstavljaju izlaz iz sloja za svaku reč. Može biti na primer 32, 100, ili čak više. Testiranjem treba proveriti šta najviše odgovara.
- input_length - dužina ulazne sekvence. Broj reči po dokumentu

Kako svi filmovi imaju opise različite dužine, radimo poravnanje. Uzimamo 200 reči iz svakog opisa. Sve opise spajamo u jedan korpus i na osnovu njega formiramo ulazne i izlazne podatke. Ulazni su sekvence dužine 20 reči, izlaz je naredna reč.

Na slici 10 prikazan je primer kreiranja ulaza i izlaza, gde je ulaz sekvenca dužine četiri reči, a izlaz je peta reč. Ulaz je uokviren plavom linijom, a izlaz je obojen u narandžasto.



Slika 10: Primer kreiranja ulaza i izlaza

3.2 Kreiranje i treniranje modela

Koristićemo dva LSTM sloja u mreži. Dodatni skriveni slojevi produbljuju mrežu, čime ovaj model više zaslužuje naziv tehnike dubokog učenja. Dubina mreže se generalno povezuje sa njihovim uspehom, pri primeni na razne probleme predviđanja.

"The success of deep neural networks is commonly attributed to the hierarchy that is introduced due to the several layers. Each layer processes some part of the task we wish to solve, and passes it on to the next. In this sense, the DNN can be seen as a processing pipeline, in which each layer solves a part of the task before passing it on to the next, until finally the last layer provides the output."

— Training and Analyzing Deep Recurrent Neural Networks, 2013 [**?**]

U okviru rada „Speech Recognition with Deep Recurrent Neural Networks“ [**?**] zaključeno je da je dubina neuronske mreže bitnija od broja neurona u okviru datog sloja.

Mreža sadrži:

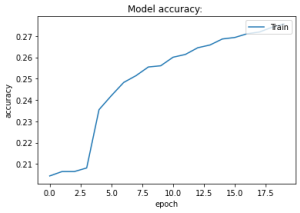
1. **Ulazni sloj**. Koristimo Embedding sloj radi pogodne reprezentacije reči.
2. **Prvi LSTM sloj**. Sadrži nekoliko stotina skrivenih neurona. O uticaju broja neurona na uspešnost mreže biće reči kasnije.
3. **Sloj izbacivanja**. Radi sprečavanja pojave preprilagođavanja. Konvergencija je sporija, ali i kvalitetnija. Generalizacija je bolja zahvaljujući tome što se u treniranju izbegava to da jedan neuron bude sam odgovoran za neke primere, s obzirom na to da prisustvo neurona nije garantovano.
4. **Drugi LSTM sloj**. Isti kao prvi.
5. **Izlazni sloj**. Koristi softmax funkciju aktivacije, koja dodeljuje verovatnoću svakom mogućem izlazu.

4 Rezultati

U ovom poglavlju analiziraćemo uspešnost rešavanja problema predikcije sledećih N reči u rečenici. Opisane su uočene ralike u ponasanju modela sa promenom broja neurona i broja epoha.

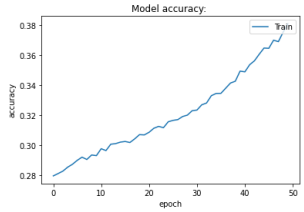
4.1 Brzina konvergencije

Slika 11 prikazuje brzinu konvergencije prilikom treniranja modela koji ima po 100 neurona u svakom LSTM sloju, dužine trajanja 20 epoha.



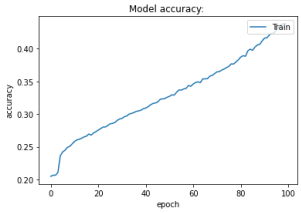
Slika 11: Grafik konvergencije u 20 epoha

Na slici 12 je prikazana brzina konvergencije tokom 50 epoha.



Slika 12: Grafik konvergencije u 50 epoha

Na slici 13 je prikazana brzina konvergencije tokom 100 epoha.



Slika 13: Grafik konvergencije u 100 epoha

4.2 Veličina mreže i broj epoha

Jedno od pitanja koje nema definitivan odgovor kada god se radi o neuronskim mrežama je veličina mreže. Drugim rečima, pitanje je koliko skrivenih neurona staviti kako bi se izbalansiralo između dobrih performansi na trening delu podataka i dovoljno dobre generalizacije. Suština ove dileme leži u balansiranju između potprilagođavanja i preprilagođavanja. Naime, ukoliko skrivenim slojevima dodelimo previše neurona, tada će model veoma dobro naučiti podatke trening skupa, ali će varijansa biti prevelika. S druge strane, previše jednostavan model nije u stanju da nauči sve zakonitosti koje vladaju u skupu koji mu je dodeljen.

Broj epoha je parametar koji definiše koliko će puta algoritam učenja raditi sa čitavim skupom trening podataka. Jedna epoha znači da je svaki

uzorak u skupu podataka za obuku imao mogućnost ažuriranja internih parametara modela. Sa porastom broja epoha smanjuje se greška modela.

Probaćemo nekoliko različitih veličina mreža i broja epoha i naći onu optimalnu.

Broj epoha	Broj neurona	Tacnost	Gubitak
20	100	27.38	5.11
50	100	38.18	4.27
100	100	43.88	3.88
20	200	27.80	4.83
50	200	38.78	3.96

Tabela 1: Varijacije modela

Tabela 1 potvrđuje da porastom broja neurona i broja epoha dobijamo bolji model. Vidimo da najveću tačnost ima model sa 100 epoha i 100 neurona kao i najmanji gubitak u odnosu na ostale testirane modele. Ovim pokazujemo da nas ne vodi samo promena broja neurona ka najboljem modelu. Potrebno je istraživati različite parametre kako bi došli do najboljeg, odnosno, onog koji će davati najsmislenije rezultate.

Rezultat za model koji ima 100 epoha i 100 neurona prikazan je slikom ispod 14. Ulazni tekst je 'The movie' i vraćeno je 50 narednih reči koje generiše naš model.

```
print(generate_words("The movie", 50, model))
```

```
The Movie And Everyone But Is Kidnapped To Be A Young Woman Who Is An Matador And Uses His Contented Hack Benet Br
eaks A Merry Strife That He Has Been Forced To Be Changed By Ghostface In The Family But Been Slaughtered By His S
traight Priest Who Is An Dominates Who Is
```

Slika 14: Rezultat 100e100n

Model vraća smislene reči za dati ulaz. U nastavku prikazane su slike za ostale varijacije modela kako bi čitalac imao više osećaja o razlikama. Notacija 100e100n označava da je reč o modelu koji ima 100 neurona u LSTM sloju i da je treniran kroz 100 epoha.

```
print(generate_words("The movie", 50, model))
```

```
The Movie And A Family And A Family And Is Has And Has And Has And Has And Has And Has And Has And Has And Has And
Has And Has And Has And Has And Has And Has And Has And Has And Has And Has And Has And Has And Has And Has And
```

Slika 15: Rezultat 20e100n

```
print(generate_words("The movie", 50, model))
```

```
The Movie And Harford Phil Wheeler In Paradise Overhearing Reptiles The Fight Of A Local Woman And Has Been Forced
To Be A Wife Technician And His Friend Who Is An Mother Who Is An Illegitimate Family Jr A J And Thief A Wife Tech
nician And Says And Says And Tells By
```

Slika 16: Rezultat 50e100n

```
print(generate_words("The movie", 50, model))
```

```
The Movie The Wife And Is Been Been A Wife And Is A Wife And Is A Wife And Is A Wife And Is A Wife And Is A Wife A
nd Is A Wife And Is A Wife And Is A Wife And Is A Wife And Is A Wife And Is
```

Slika 17: Rezultat 20e200n

```
print(generate_words("The movie", 50, model))
```

```
The Movie And Harford Wife Man A Young Man Living In A Man Of Is Arrive To Be A Young Woman Who Is Been Forced To
Be A Young Woman Who Is Been Forced To Be A Young Woman Who Is Been Forced To Be A Young Woman Who Is Been Forced
```

Slika 18: Rezultat 50e200n

5 Zaključak

Zahvaljujući tome što modeluju prelaze iz stanja u stanje, umesto direktne zavisnosti izlaza od celokupnog ulaza, rekurentne mreže su u stanju da predstave vrlo duboke strukture zavisnosti i da obrađuju sekvence promenljive, a velike dužine. Zahvaljujući LSTM jedinicama, takođe su u stanju da ublaže problem nestajućih gradijenata i da modeluju dugoročne zavisnosti.

Ipak, zbog kompleksnosti podataka koje modeluju i broja parametara koje LSTM jedinice imaju, nije ih lako obučavati (npr. jedna od poteškoća je haotično ponašanje u slučaju velikih apsolutnih vrednosti parametara). Poznato je da je njihovo obučavanje teže hardverski ubrzati zbog visokog obima memorijskog protoka i sekvencijalne prirode izračunavanja (dok hardversko ubrzanje počiva na paralelizaciji), što je ključ efikasnog obučavanja neuronskih mreža.

5.1 Dalji razvoj

Svakako, prikazano rešenje nije najbolje. Postoje razni načini kojima se LSTM neuronska mreža moze poboljšati za ovaj problem a neki od saveta za dalji rad dati su u nastavku.

- Povećajte broj epoha na više stotina, ili čak stotine stotina.
- Eksperimentišite sa broj neurona kao i brojem slojeva mreže.
- Razmisliti o slojevima napustanja, odnosno, zaboravljanja i procentima za iste.
- Menjati parametre, uočiti kako utiču na rezultat i jedni na druge.