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

15. 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 [4]. Automatsko generisanje teksta može biti implementirano i na nivou karaktera, što je opisano u radu [6].

LSTM mreže koriste se za razne probleme nad sekvencijalnim podacima. Tako imamo radove koji se bave automatskim generisanjem reči pesama [5], ili čak automatskim komponovanjem muzike [2].

2 Rekurentne neuronske mreže

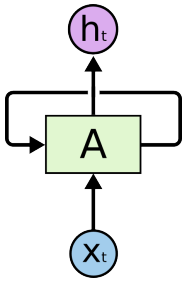
Rekurentne neuronske mreže (eng. Recurrent Neural Networks - RNN) predstavljaju arhitekturu mreža specijalizovanu za obradu sekvencijalnih podataka, poput rečenica prirodnog jezika i vremenskih serija. Sekvence nameću važnost redosleda zapažanja podataka, kako prilikom treniranja modela, tako i prilikom 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 elementiam 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.

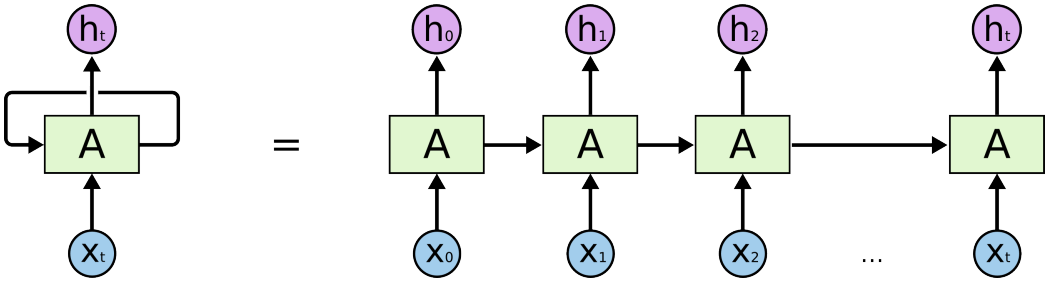
2.1 Arhitektura RNN

Rekurentne neuronske mreže sadrže petlje koje obezbeđuju čuvanje informacija [1].



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.

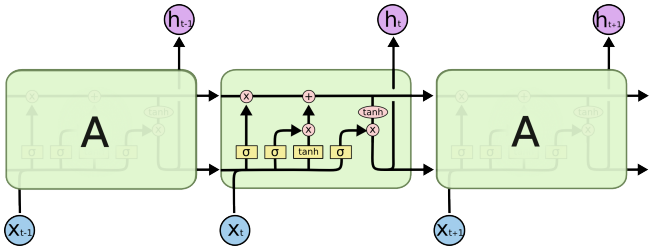
Drugi problem se odnosi na dugoročno čuvanje informacije i modelovanje dugoročnih zavisnosti u podacima. Kako se skriveno stanje u svakom koraku dobija linearnom kombinacijom prethodnog stanja i ulaza, doprinos starijih ulaza se brzo gubi pod uticajem novih. Dugoročno čuvanje relevantnih informacija 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. [3]

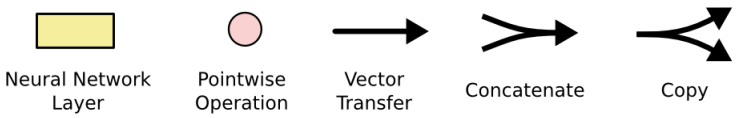
2.3 LSTM neuronske mreže

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).



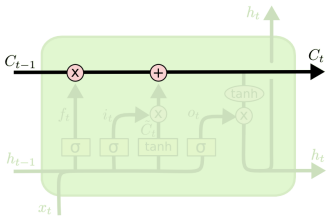
Slika 3: Prikaz lanca LSTM mreže

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

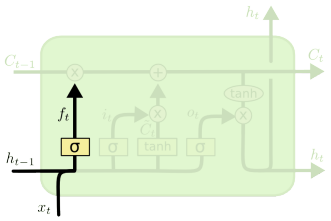
Osnovna ideja LSTM-a je postojanje takozvane ćelije koja čuva skriveno stanje, uz kontrolu pisanja, čitanja i zaboravljanja. Na slici 5 je to horizontalna linija na vrhu dijagrama.



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.

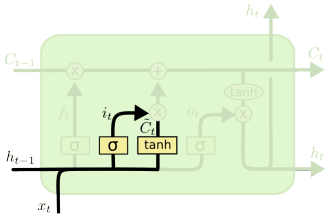
Prvi korak u našoj LSTM mreži je odabir informacija koje će biti obrisane iz stanja ćelije. Kapija koja kontroliše ovu operaciju naziva se „kapija zaboravljanja“. Ovaj sloj koristi sigmoidnu aktivacionu funkciju, čiji je izlaz interval brojeva između 0 i 1. Vrednost 1 znači da se potpuno zadržava sadržaj,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

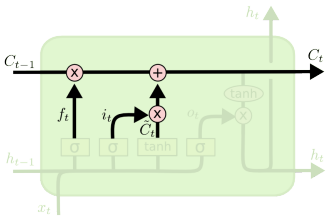
Sledeći korak predstavlja odlučivanje o tome koje nove informacije ćemo dodati u stanje ćelije. Sastoji se iz dva dela. Prvo, sloj koji predstavlja „ulaznu kapiju“ odlučuje koje će vrednosti ažurirati. Zatim naredni sloj, čija je aktivaciona funkcija tangens hiperbolički, pravi vektor novih vrednosti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

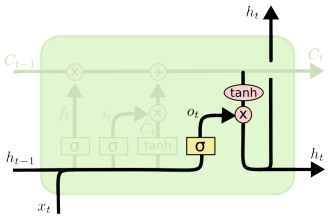
Sada prethodno stanje ćelije C_{t-1} ažuriramo u novo C_t. Pomnožimo staro stanje sa f_t, time zaboravljajući ono što smo odlučili da zaboravimo. Zatim dodamo $i_t \times \tilde{C}_t$. Ovo su potencijalne nove vrednosti, skalirane za onoliko koliko smo odlučili da ažuriramo svaku vrednost stanja (Slika 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 [7]. .

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
- Generisanje teksta

3.1 Preprocesiranje ulaznih podataka

Ulazni podaci su preuzeti iz baze podataka koja sadrži radnje filmova. Na slučajan način biramo 200 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.

3.2 Kreiranje i treniranje modela

Za trening mreže koristimo tokenizovane reči, sakupljene iz sadržaja 200 filmova. Broj tokena iznosi 10806. Ulazni podaci su sekvence dužine 20 reči, a izlazni naredne reči. Mreža se sastoji iz ulaznog sloja (eng. Embedding layer), LSTM sloja, i izlaznog sloja. Izlazni sloj koristi softmax funkciju aktivacije. Funkcija gubitka je cross-entropy. Trening se izvršava u 100 epoha. Radi efikasnijeg treninga, koristimo tehniku slučajnog odbacivanja pojedinih veza (dropout) i Adam optimizator.

```
model = Sequential()
model.add(Embedding(vocab_size+1, 10, input_length = trainX.shape[1]))
model.add(LSTM(256))
model.add(Dropout(0.1))
model.add(Dense(trainy.shape[1], activation = 'softmax'))
```

Slika 10: Konfiguracija modela

4 Zaključak

Literatura

[1] Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] Keunwoo Choi, George Fazekas, and Mark Sandler. Text-based lstm networks for automatic music composition. *arXiv preprint arXiv:1604.05358*, 2016.

[3] Mladen Nikolić i Anđelka Zečević. *Mašinsko učenje*. 2019.

[4] D Pawade, A Sakhapara, M Jain, N Jain, and K Gada. Story scrambler–automatic text generation using word level rnn-lstm. *International Journal of Information Technology and Computer Science (IJITCS)*, 10(6):44–53, 2018.

[5] Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924, 2015.

[6] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1017–1024, 2011.

[7] Ziwen Wang, Jie Wang, Haiqian Gu, Fei Su, and Bojin Zhuang. Automatic conditional generation of personalized social media short texts. In *Pacific Rim International Conference on Artificial Intelligence*, pages 56–63. Springer, 2018.

A Dodatak