Generisanje teksta - LSTM od nule

Ivan Pop-Jovanov

Septembar 2022

Sažetak

U ovom radu prikazujem implementaciju neuronske mreže trenirane za generisanje teksta. Implementirao sam u programskom jeziku Python biblioteku za rad sa sekvencijalnim neuronskim mrežama, kao i podršku za potpuno povezani sloj, rekurentni sloj, i LSTM sloj. Krajnji model je sposoban da nauči osnovna pravila prirodnog jezika, i dostiže preciznost predviđanja uporedivu sa istim modelom implementiranim u Keras biblioteci.

Sadržaj

1 Uvod

Natural Language Processing (NLP) je oblast na preseku lingvistike i veštačke inteligencije čija je popularnost eksponencijalno porasla u zadnjih dvadesetak godina.

Modeli jezika su našli mnogobrojne slučajeve upotrebe, kao što su provera pravopisa, detekcija neželjene elektronske pošte, automatsko odgovaranje na korisnička pitanja, klasifikacija emocija teksta, kao i mnoge druge.

U većini sučajeva, zadatak je da model na osnovu sekvence tokena jezika predvidi raspodelu verovatnoća narednog tokena. Tokeni mogu da budu pojedinačni karakteri (character based approach), kao i cele reči (word based approach). U ovom radu predstavljam implementaciju zasnovanu na predviđanju narednog karaktera u tekstu, mada je bitno naglasiti da su modeli zasnovani na vektorima reči danas de facto standard zbog povećane efikasnosti treniranja, kao i generalno boljih rezultata.

Prvi NLP modeli su koristili jednostavne feed-forward mreže, ali zbog sekvencijalne prirode problema, rekurentne mreže su dugi niz godina bile glavni fokus istraživanja u oblasti jezičkog modelovanja. Najveći doprinos je napravilo uvođenje LSTM sloja[1], koji je omogućio modelima da donose odluke o predviđanju narednog tokena na osnovu tokena koji se nalaze dosta ranije u tekstu, efektivno simulirajući neku vrstu kratkotrajnog pamćenja.

Opšte je poznato da su rekurzivni modeli jako spori i teški za treniranje. Sledeći korak i veliki napredak u oblasti modelovanja jezika su dopreneli mehanizmi pažnje i novonastali transformer modeli[2]. Ovaj pristup odbacuje sekvencijalnu obradu ulaza, time dobivši na paralelizaciji i skalabilnosti, kao i iznenađujućim napretkom u kvalitetu rezultata. U ovom radu prikazujem samo rekurzivne modele, iako su oni danas relativno zastareli.

2 Opis rešenja

Ovo poglavlje je podeljeno na tri celine. U prvoj celini ću ukratko objasniti mogućnosti i ograničenja implementirane biblioteke za rad sa neuronskim mrežama, u drugoj celini malo dublji pogled u implementaciju LSTM sloja, i na kraju ću opisati konkretan model i način treniranja u trećoj celini.

2.1 Biblioteka

Za potrebe ovog rada, implementirao sam minimalnu biblioteku za rad sa sekvencijalnim neuronskim mrežama u programskom jeziku Python. Interfejs biblioteke je dizajniran da podseća što je više moguće na interfejs biblioteke Keras, iako u pozadini ne koristi ni Tensorflow ni PyTorch. Sva matrična izračunavanja se vrše na procesoru korišćenjem NumPy biblioteke, dok rad sa grafičkim karticama nije podržan. U procesu implementacije, akcenat je stavljen na jednostavnost koda mnogo više nego na optimizaciju.

2.1.1 Aktivacione funkcije

Svaka aktivaciona funkcija ima isti interfejs koji podržava propagaciju u napred i u nazad. Implementacija novih aktivacionih funkcija se svodi na implementaciju te dve funkcije.

Podržano je pet aktivacionih funkcija:

1. Linearna aktivaciona funkcija
2. ReLU aktivaciona funkcija
3. Tanh aktivaciona funkcija
4. Sigmoidna aktivaciona funkcija
5. Softmax aktivaciona funkcija

2.1.2 Slojevi neurona

Kao i kod aktivacionih funkcija, interfejs sloja se svodi na dve funkcije, propagacija u napred i propagacija u nazad. Pri kreiranju sloja prosleđujemo ulaznu i izlaznu dimenziju sloja, kao i aktivacionu funkciju.

Podržana su tri sloja neurona:

1. Gusto povezani sloj
2. Jednostavni rekurzivni sloj
3. LSTM sloj

2.1.3 Funkcije greške

Funkcija greške je implementirana kao klasa čiji interfejs predstavlja dve funkcije, funkciju za računanje vrednosti greške i funkciju za računanje gradijenta.

Podržane su dve funkcije greške:

1. Srednje kvadratna greška
2. Kategorička unakrsna entropija

2.1.4 Algoritmi za optimizaciju

Pri instanciranju objekta klase optimizatora, podržana je većina standarnih parametara procesa optimizacije, kao što su broj epoha, brzina učenja, funkcija greške, batch size, itd.

Podržana su dva algoritma za optimizaciju:

1. Stohastički gradijentni spust
2. Adam optimizacioni algoritam

2.1.5 Sekvencijalni model

Glavna klasa biblioteke je klasa Sequential, koja podržava dodavanje slojeva, treniranje korišćenjem instance klase optimizatora, kao i predikciju.

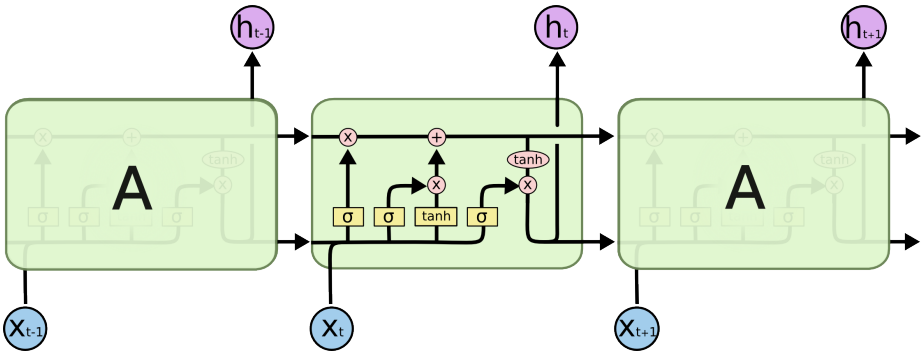
Listing 1: Primer korišćenja biblioteke

```
X = np.arange(-1,1,0.01).reshape((-1,1))
y = np.sin(X*3*np.pi)

optimizer = OptimizerAdam(
    learning_rate=0.05,
    n_epochs=1000,
    batch_size = 16,
    loss=LossMSE(),
    printEvery=10)

model = Sequential(optimizer)
model.add(Dense(1,10,activation = ActivationTanh()))
model.add(Dense(10,1,activation = ActivationLinear()))

model.train(X,y)

plt.plot(X,y)
plt.plot(X,model.predict(X))
```

2.2 Long Short-Term Memory

LSTM[1] sloj je vrsta rekurentnog neuronskog sloja koji je nastao sa ciljem da reši određene probleme koji su nastajali u klasičnim RNN mrežama. Iako je glavna motivacija iza rekurentnih mreža činjenica da mogu da pamte informacije dok prolaze kroz sekvencu, u praksi se primećuje da se jednostavne RNN mreže muče da nauče kako da čuvaju informacije koje su videle dosta ranije u sekvenci.

Pokazalo se da LSTM mreže nemaju tu vrstu problema. Glavna razlika je što LSTM mreže čuvaju svoju memoriju u nečemu što se zove *cell state*, obično označeno sa C ili C_t, i uče da preko skalarnih proizvoda odlučuju koji deo te memorije treba zadržati a koji ne.



Slika 1: Standardna verzija LSTM sloja,
preuzeto sa http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Kao što se vidi na slici 1, u trenutku t, sloj uz X_t kao ulaz dobija i prethodni izlaz h_{t-1}, kao i prethodni cell state C_{t-1}. Prvo se X_t i h_{t-1} spajaju i zajedno propuštaju kroz četiri gusto povezana sloja (označeno žutom bojom na slici) a nakon toga izlaz tih slojeva kroz niz vektorskih operacija menja cell state (označeno roze bojom na slici).

Postoje različite konfiguracije unutrašnje strukture LSTM sloja specifično odabrane za probleme koje rešavaju. Konkretna implementacija prikazana na slici 1 se smatra standardnom, i nju sam pratio u ovom radu. Za više informacija pogledati izvorni kod.

2.3 Model

Model je treniran nad *The Lovecraft Corpus*, skupom kratkih priča autora H. P. Lovecraft, preuzetog sa https://github.com/vilmibm/lovecraftcorpus.

Ovaj skup ima 2890396 karaktera ukupno, odnosno 93 različitih. Nije vršeno nikakvo predprocesiranje ili uklanjanje karaktera.

Model ima dva skrivena sloja, jedan LSTM veličine 256, a drugi gusto povezani veličine 256. Brzina učenja je 10^{-3}, dužina sekvence je 128, a funkcija greške je kategorička unakrsna entropija.

Za potrebe poređenja rezultata, implementirao sam isti model i u Keras biblioteci. U nastavku se nalaze obe implementacije.

Listing 2: Model - Implementacija u sopstvenoj biblioteci

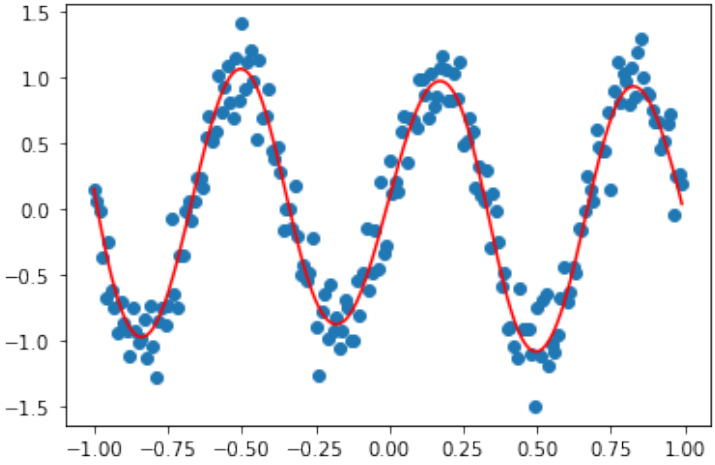
```
optimizer = OptimizerAdam(
  learning_rate=0.001,
  batch_size = 128,
  loss=LossCrossEntropy())

hidden_size = 256

model = Sequential(optimizer)
model.add(LSTM(
  vocab_size,
  hidden_size,
  activation = ActivationLinear()))
model.add(Dense(
  hidden_size,
  hidden_size,
  activation = ActivationTanh()))
model.add(Dense(
  hidden_size,
  vocab_size,
  activation = ActivationSoftmax()))
```

Listing 3: Model - Implementacija u Keras biblioteci

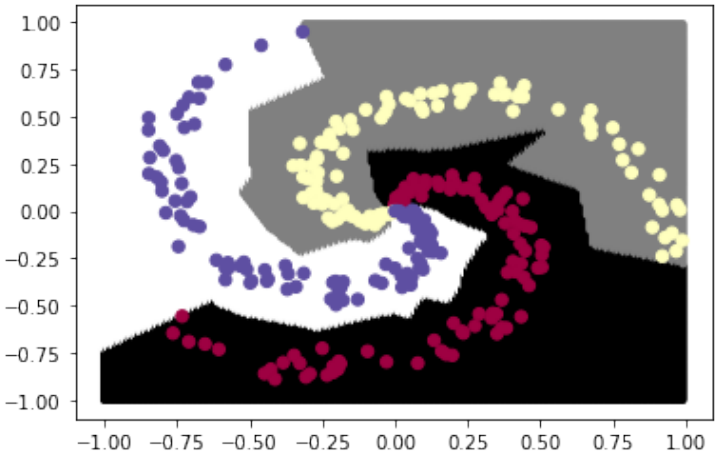
```
hidden_size = 256

model = Sequential()
model.add(LSTM(
    hidden_size,
    input_shape=(X.shape[1], X.shape[2])))
model.add(Dense(
    hidden_size,
    activation='tanh'))
model.add(Dense(
    y.shape[1],
    activation='softmax'))
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam')
```

3 Eksperimentalni rezultati

Pre nego što sam pokrenuo trening glavnog modela, prvo sam testirao implementaciju biblioteke nad problemima regresije i klasifikacije. Modeli daju zadovoljavajuće rezultate, kao što se može videti na slikama 2 i 3.



Slika 2: Regresija



Slika 3: Klasifikacija

Sledeća stvar koju sam testirao je trening generisanja teksta na jako malom ulaznom skupu. Specifično, model je treniran na rečenici "The quick brown fox jumps over the lazy dog.", koja ima 45 karaktera ukupno, odnosno 29 različitih. Kao što je i očekivano model je brzo konvergirao jer je uspeo da nauči rečenicu napamet.

Ako zahtevamo od modela da predvidi narednih 100 karaktera nakon početne niske "T", dobijamo "The quick brown fox jumps over the lazy dog. ox te fox tazy dog. over tazy ug. ox fthe lazy dog. ox t".

Za početnu nisku "q", dobijamo "quick. own fox jump the lazy dog. ox te

fox the lazy dog. ox the fo.. over the lazy dog. ox te e lazy".

Za početnu nisku "jump", dobijamo "jumps over the lazy iick brown fox jumps over the lazy dog. ox te fox the lazy dog. ox the fog. ox the f".

Ovo su otprilike i očekivani rezultati, model je naučio kako da završi rečenicu na kojoj je treniran, ali očigledno nije moguće da iz ovako malog ulaznog skupa nauči bilo kakva pravila prirodnog jezika.

Nakon toga, model sam trenirao na način opisan u sekciji 2.3 dok funkcija gubitka nije konvergirala do vrednosti oko 0.28 i tada sam prekinuo. Za početnu nisku "the", model vraća narednih 500 karaktera:

"the moong has somewhat oo hir woing ho wounh saint in tn hised tide aiseso homedhinealy. He opent ahat hishing hosks but tir resegas . Wuths he wonhing whan he wished yome had rome ent conessine and Monersnt entiess and Monk sing work, but the donn, hunlell, Cunseell an tranlenglangs an to frig, he wrdion to Mre hn oone and yry neng dook accinginaling in the gornt in whise saine iistigest ano monhid woniing and yo f Frie, hun equenteainglon hise and honement on tia were saditanle and made sigatlecc"

Primetimo da je model uspeo da nauči određena pravila engleskog jezika. Nakon tačke stavlja razmak i veliko slovo, nakon zareza razmak. Veliko slovo koristi samo na početku reči, i koristi česte konstrukte, kao što su "and", "the", "but the", "in the", kao i zamenicu trećeg lica muškog roda "he". Pojavljuje se dosta pravih reči engleskog jezika, kao somewhat, saint, wished, sing, work, made, ali i dosta reči koje ne postoje, mada idalje prate stil engleskog jezika, kao hishing, accinginaling, equenteainglon, honement.

Model treniran pomoću Keras biblioteke daje slične rezultate, jedina razlika je što sam, da bih ubrzao treniranje, pretvorio sve karaktere u mala slova, pa se zato u primeru ne pojavljuju velika slova:

"the stonice sey was sas mup inthes in a sooad efef shfes aatorees tha wokdiss sosn, whel i gas wponiarin thi oopg cdl wiv toeds,at i day when the haster if i cad oort toach in my seritord sr merited toad ant a sastet oe tae thedo thane; tha inknn andert and cay b ingesd po isrfintef tial a doute netea tor oede budder thet the eese uft put wca an thene ctom the derranne arulati;ral e toonstoon as as atc lenhleyson of she vonlds oe the holt th cgn at the hate in the ceekes. and she oase lastne and of"

Iznenađujuće, model u Keras biblioteci je bio dosta teži za treniranje. Duže mu je trebalo da konvergira i često je kao izlaz ispisivao dugačke niske u kojima samo ponavlja česte reči kao "and", ili "the". U oba modela su korišćeni isti parametri, uključujući i istu brzinu učenja, tako da su potrebni dalji eksperimenti da zaključim razlog iza različitih rezultata.

4 Zaključak

Očekivano, rezultati nisu ni blizu rezultatima koji se pojavljuju u modernim istraživanjima na temu jezičkog modelovanja. Da bi dostigli mogućnost generisanja smislenog teksta, potrebno je napraviti model sa znatno većim brojem parametara, i trenirati ga na velikim skupom podataka. Na primer, jezički model GPT-3 ima 175 milijardi parametara i treniran je na 45 terabajta teksta[3].

Osim skaliranja modela, veliki napredak bi se mogao ostvariti i korišćenjem pristupa vektora reči, gde umesto predviđanja karakter po karakter predviđamo reč po reč. Takođe, moja implementacija ne podržava batch pristup, što čini konvergenciju veoma sporom. Dodatno, treniranje bi se moglo ubrzati i uvođenjem predprocesiranja teksta za smanjenje broja različitih karaktera između kojih model mora da bira, kao na primer uklanjanje interpunkcije ili pretvaranje velikih slova u mala.

Literatura

[1] Hochreiter, Sepp, and Jürgen Schmidhuber. *"Long short-term memory."* Neural computation 9.8 (1997): 1735-1780.

[2] Vaswani, Ashish, et al. *"Attention is all you need."* Advances in neural information processing systems 30 (2017).

[3] Brown, Tom, et al. *"Language models are few-shot learners."* Advances in neural information processing systems 33 (2020): 1877-1901.