# Rekurentne neuronske mreže

**Rajić, Ivana**

**Master's thesis / Diplomski rad**

**2024**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:217:923477

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

Ivana Rajić

# REKURENTNE NEURONSKE MREŽE

Diplomski rad

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET
FIZIČKI ODSJEK

INTEGRIRANI PREDDIPLOMSKI I DIPLOMSKI SVEUČILIŠNI STUDIJ
FIZIKA I INFORMATIKA; SMJER NASTAVNIČKI

**Ivana Rajić**

Diplomski rad

# Rekurentne neuronske mreže

Voditelj diplomskog rada: Izv. prof. dr. sc. Maro Cvitan

Ocjena diplomskog rada: _____

Povjerenstvo: 1. _____

2. _____

3. _____

Datum polaganja: _____

Zagreb, 2024.

# Sažetak

U ovom radu istražuju se rekurentne neuronske mreže (RNN) i njihova primjena u analizi sekvencijalnih podataka. Poseban naglasak stavljen je na arhitekturu Long Short-Term Memory (LSTM) koja se koristi za rješavanje problema nestajanja i eksplodiranja gradijenata u standardnim RNN modelima. Rad također opisuje poznata svojstva i izazove s kojima se susreću RNN, LSTM kao i nedavno predložena xLSTM arhitektura. U završnom dijelu implementirane su RNN i LSTM mreže za generiranje teksta te su rezultati analizirani kako bi se usporedile ove dvije arhitekture.

Ključne riječi: Rekurentna neuronska mreža, Long short term memory, neuron, trening, generiranje teksta, gubitak, funkcija gubitka, aktivacijska funkcija, xLSTM

# Recurent neural networks

## Abstract

This thesis explores recurrent neural networks (RNN) and their application in the analysis of sequential data. Special emphasis is placed on the Long Short-Term Memory (LSTM) architecture, which is used to address the issues of vanishing and exploding gradients in standard RNN models. The paper also describes known properties and challenges encountered by RNNs, LSTMs, as well as the recently proposed xLSTM architecture. In the final section, RNN and LSTM networks are implemented for text generation and the results are analyzed to compare these two architectures.

Keywords: Recurent neural network, Long short term memory, neuron, training, text generating, loss, loss function, activation function, xLSTM

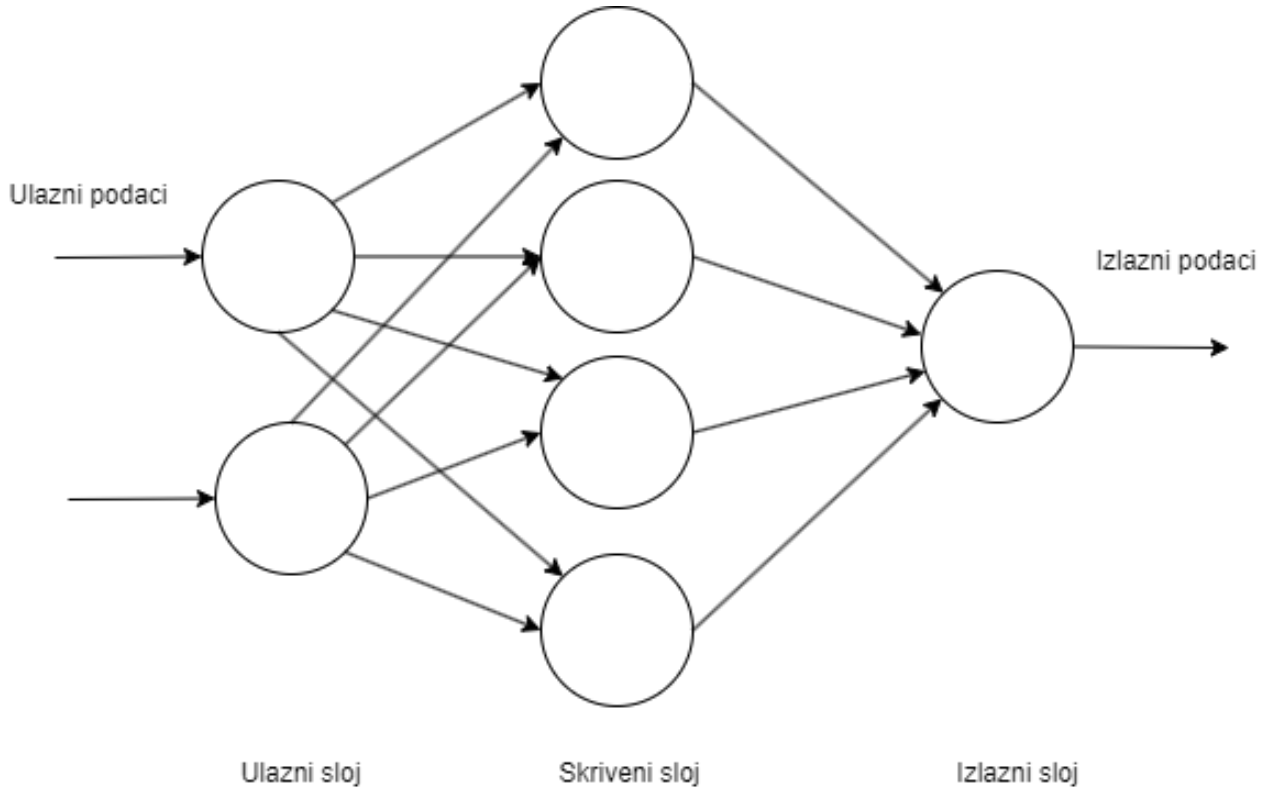
# Sadržaj

# 1 Uvod

## *1.1 Neuronske mreže*

Neuronske mreže predstavljaju tip algoritma za strojno učenje, nadahnute strukturom i funkcijom ljudskog mozga. Glavni cilj neuronskih mreža je, poput mozga, prepoznavanje veza među podacima i učenje složenih obrazaca, kako bi se na temelju stečenog znanja mogle donositi predikcije i odluke. Koncept neuronskih mreža prvi su spomenuli Warren McCulloch i Walter Pitts u svom radu iz 1943. godine [1].

Shematski, neuronske mreže sastoje se od međusobno povezanih jedinica. Ovisno o kontekstu te se jedinice (ili određene podjedinice) nazivaju neuroni. Treba napomenuti da je slika 1.1 okvirna te da ćemo preciznije definirati pojmove u sljedećim poglavljima. Prema [2], svaka jedinica predstavlja strukturu koja prima jedan ili više ulaznih podataka kojima dodjeljuje neke brojčane vrijednosti. Više takvih jedinica organizirano je u tri sloja: ulazni sloj koji prima podatke, skriveni sloj odgovoran za učenje i prepoznavanje složenih obrazaca, te izlazni sloj koji generira konačne izlazne podatke.




Slika 1.1: Dijagram neuronske mreže

Na slici 1.1 jedinice (neuroni) su prikazane u obliku krugova. Važno je naglasiti da je ovo pojednostavljen prikaz, budući da broj jedinica u svim slojevima može varirati, kao i broj skrivenih slojeva. Sve neuronske mreže imaju ovakvu strukturu, ali postoji više različitih tipova neuronskih mreža, prilagođenih specifičnim zadacima i vrstama podataka [4]:

1. Unaprijedna neuronska mreža (eng. *Feedforward Neural Network, FNN*) [5] predstavlja najjednostavniji oblik neuronske mreže, gdje informacije teku u jednom smjeru, od ulaznog do izlaznog sloja. Najčešće se koristi za zadatke klasifikacije i regresije.

2. Konvoluirana neuronska mreža (eng. *Convolutional Neural Network*, CNN) [6] specijalizirana je za obradu slika i videa. Izuzetno je uspješna u zadacima prepoznavanja objekata i u zadacima segmentacije slika.

3. Rekurentna neuronska mreža (eng. *Recurent neural network*, RNN) [4] dizajnirana je za rad sa sekvencijalnim podacima, gdje izlaz u svakom vremenskom koraku ovisi o prethodnom. Detaljnije će biti objašnjena u nastavku rada.

4. Generativna adversarijalna mreža (eng. *Generative Adversarial Network, GAN*) [7] sastoji se od dvije mreže, generatora i diskriminatora, koje se treniraju u natjecateljskom okruženju kako bi generirale realistične podatke. Ove mreže se koriste za generiranje umjetničkih djela, poput tekstova i slika.

5. Transformativna neuronska mreža (eng. *Tansformative neural network*) [8] koristi se za zadatke poput strojnog prevođenja i modeliranja jezika. Trenutačno se široko primjenjuje u komercijalnim sustavima kao temelj za modele GPT-a (eng. *Generative Pre-trained Transformer*).

## *1.2 Treniranje neuronske mreže*

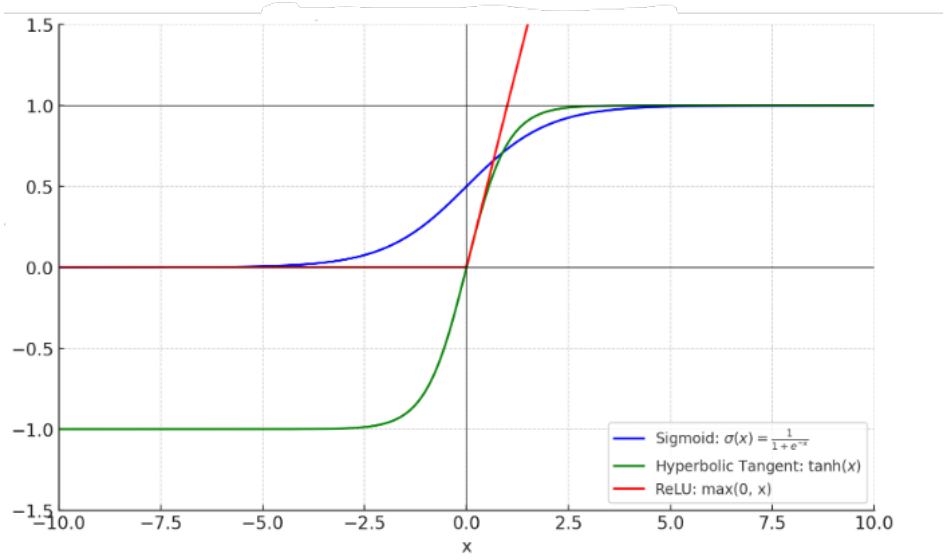
Kako bi neuronske mreže mogle generirati izlazne podatke, potrebno ih je trenirati. Trening neuronske mreže uključuje proces prilagodbe težina između neurona kako bi se smanjila razlika između predviđenih i stvarnih izlaza. Težine (eng. *Weights*) predstavljaju numeričke vrijednosti. Tijekom treninga, mreža prilagođava te težine kako bi poboljšala točnost svojih predikcija. Što je težina veća ili manja, to je

jači ili slabiji utjecaj ulaznog podatka na izlaz. Proces treninga odvija se s pomoću algoritma unazadne propagacije (eng. *Backpropagation*), pri čemu mreža izračunava gradijent funkcije gubitka uzimajući u obzir parametre, odnosno težine i pristranosti, te koristi optimizacijske algoritme za ažuriranje tih parametara. Funkcija gubitka mjeri razliku između predviđenih i stvarnih vrijednosti modela, a rezultati koji proizlaze iz njezine evaluacije koriste se za ažuriranje težina mreže, što je ključan korak u procesu optimizacije. Funkcija gubitka još se naziva i funkcija greške (eng. *Error function*) i funkcija troška (eng. *cost function*). Cilj je minimizirati rezultat ove funkcije. Gubitak (eng. *Loss*) predstavlja rezultat funkcije gubitka - broj. Sve veličine kao i sam proces unazadne propagacije bit će matematički opisani u nastavku rada.

Jedan od ključnih elemenata neuronskih mreža su aktivacijske funkcije. One unose nelinearnost u mrežu, omogućavajući joj da nauči složene odnose između podataka. Aktivacijska funkcija se obično označava simbolom $\sigma$.

$$\sigma : \mathbb{R}^n \to \mathbb{R}^n \tag{1.1}$$

Neke od često korištenih aktivacijskih funkcija [4] su sigmoid, ReLu (eng. *Rectified linear unit*), softmax te tangens hiperbolni .




Slika 1.2: Grafički prikaz aktivacijskih funkcija

Sigmoidna funkcija definira se kao

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

$$\sigma(x) = \begin{cases} -1 & \text{za } x < 0, \\ 0 & \text{za } x = 0, \\ 1 & \text{za } x > 0. \end{cases} \tag{1.3}$$

Sigmoidna funkcija često se koristi u problemima koji zahtijevaju binarnu klasifikaciju.

ReLU definira se kao

$$ReLU(x) = max(0, x) = \begin{cases} x & \text{ako } x \geq 0, \\ 0 & \text{za } x < 0 \end{cases} \tag{1.4}$$

te se koristi u skrivenim stanjima zbog svoje jednostavnosti i sposobnosti da umanji problem nestajućeg gradijenta. O problemu nestajućeg gradijenta će više riječi biti u poglavlju 2.5.

U ovom radu kasnije će se koristiti softmax funkcija [4] kao aktivacijska funkcija. Definira se kao funkcija $\sigma : \mathbb{R}^K \to (0,1)^K$, gdje $K \geq 1$, koja uzima vektor $\mathbf{z} = (z_1, ...., z_K) \in \mathbb{R}^K$ te kao rezultat daje raspodjelu vjerojatnosti $\sigma(\mathbf{z}) \in (0,1)^K$

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{1.5}$$

Ova aktivacijska funkcija najčešće se koristi u zadacima koji zahtijevaju višerazrednu klasifikaciju. 2D vizualizacija softmax funkcije dostupna je na [13].

Treniranje neuronskih mreža odvija se kroz tzv. epohe. Epohe su hiperparametar mreže. To znači da se zadaju kao fiksna vrijednost na početku treninga. Tijekom jedne epohe, mreža obrađuje cijeli skup podataka za treniranje, bilo odjednom ili podijeljeno u manje dijelove. Proces treniranja obično se provodi kroz više epoha, čime se omogućuje da mreža više puta prođe kroz sve podatke i bolje prilagodi svoje parametre kako bi se poboljšala točnost izlaznih rezultata. Na primjer, ako imamo skup podataka za treniranje koji sadrži 1000 uzoraka, tijekom jedne epohe mreža će proći kroz sve te uzorke jednom. Nakon deset epoha, model će pregledati i učiti na istom skupu podataka deset puta. Prevelik broj epoha može dovesti do problema prenaučenosti [4], gdje model postaje previše prilagođen skupu podataka za treniranje, a time gubi sposobnost generalizacije na novim podacima. S druge strane, ako broj

epoha bude premalen, model možda neće imati dovoljno vremena da nauči dovoljno informacija iz skupa podataka, što može rezultirati lošijom točnošću predviđanja.

# 2 Rekurentne neuronske mreže

Rekurentne neuronske mreže (RNN) jedne su od najistaknutijih tipova mreža u području dubokog učenja, posebno se koriste analizi i obradi sekvencijalnih podataka. Za razliku od klasičnih neuronskih mreža, RNN posjeduju memorijsku komponentu koja im omogućuje prepoznavanje i iskorištavanje ovisnosti među podacima [4]. Zbog toga su idealne za zadatke koji uključuju sekvencijalne podatke, kao što su procesiranje prirodnog jezika, modeliranje jezika, generiranje teksta, prepoznavanje entiteta i strojno prevođenje [11]. Njihova sposobnost da prepoznaju kontekstualne ovisnosti čini ih prikladnima za generiranje koherentnog teksta na određenu temu. Osim u obradi jezika, rekurentne mreže su također vrlo uspješne u zadacima poput prepoznavanja i sinteze govora te generiranja glazbe [12]. Njihova sposobnost sekvencijalnog procesiranja audio signala omogućuje im da prepoznaju vremenske karakteristike govora i glazbe, čime mogu točno generirati transkripte. RNN se također koriste za analizu i predviđanje vremenskih serija u područjima poput financija, vremenske prognoze, predviđanja potražnje za energijom i predviđanja kretanja burze [10]. Osim toga, rekurentne neuronske mreže nalaze primjenu i u analizi slika i videa, kao i u podržanom učenju (eng. Reinforcement learning) [4], gdje omogućuju modelima da uče na temelju interakcija s okolinom.

## *2.1 Arhitektura*

Rekurentna neuronska mreža s $H \in \mathbb{N}$ neurona je skup funkcija definiranih od $t = 1$ do $t = t_0$ za ulazne vektore $(x^{(1)}, ...., x^{(t_0)}), x^{(t)} \in \mathbb{R}^n$ , gdje je je $a^{(t)} \in \mathbb{R}^H$ aktivacijski vektor,

$$a^{(t)} = b_h + W_{hh}h^{(t-1)} + W_{hx}x^{(t)} \tag{2.1}$$

$t_0$ označava broj vremenskih koraka, dok je H jednak veličini skrivenog stanja.
Skriveno stanje $h^{(t)} \in \mathbb{R}^H$ definira se kao:

$$h^{(t)} = \tau_h(a^{(t)}) \tag{2.2}$$

$h^{(2)}, ..., h^{(t_0)}) \in \mathbb{R}^{t_0 \times H}$ služe kao memorijska komponenta koja zadržava informaciju o prijašnjim unosima i utječe na buduće.

$o^{(t)} \in \mathbb{R}^p$ izlazni vektor,

$$o^{(t)} = b_y + W_{yh} h^{(t)} \tag{2.3}$$

i $\hat{y}^{(t)} \in \mathbb{R}^p$ predviđeni izlazni vektor

$$\hat{y}^{(t)} = \tau_y(o^{(t)}) \tag{2.4}$$

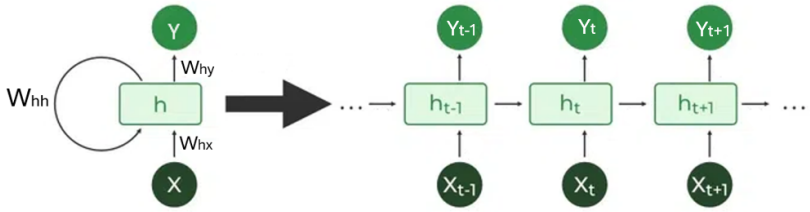
tada

$$RNN_{W_{yh},W_{xh},W_{hh},b_h,b_y,\tau_h,\tau_y,h^0}((x^{(0)}, ...., x^{(t_0)})) := (\hat{y}^{(1)}, ...., \hat{y}^{(t_0)}) \tag{2.5}$$

gdje su $W_{hx} \in \mathbb{R}^{H \times n}$, $W_{yh} \in \mathbb{R}^{p \times H}$ i $W_{hh} \in \mathbb{R}^{H \times H}$ težinske matrice (eng. *weight matrices*), $b_h \in \mathbb{R}^H$ i $b_y \in \mathbb{R}^p$ su vektori pristranosti (eng. *bias vector*), $\tau_h : \mathbb{R} \to \mathbb{R}$ i $\tau_y : \mathbb{R} \to \mathbb{R}$ aktivacijske funkcije, a $h^{(0)}$ inicijalno skriveno stanje. $o^{(t)}$ predstavlja izlazne vrijednosti koje generira mreža za vremenski korak t, $y^{(t)}$ predstavlja stvarnu vrijednost koju model pokušava predvidjeti. $\hat{y}^{(t)} \in \mathbb{R}^p$ označava predviđeni izlaz modela nakon prolaska kroz sve transformacije i aktivacije. U [3] je pokazano da navedene definicije odgovaraju određenim diskretiziranim diferencijalnim jednadžbama s kašnjenjem.

Težinske matrice predstavljaju ključne parametre u neuronskoj mreži koje se tijekom procesa treniranja prilagođavaju. Na početku treniranja, težinske matrice mogu biti inicijalizirane na različite načine, poput matrice identiteta ili matrice s nasumično zadanim vrijednostima, ovisno o pristupu koji se koristi. Aktivacijske funkcije, koje su već spomenute, uključuju tangens hiperbolni ili sigmoidnu funkciju. Ove funkcije unose nelinearnost u model, omogućujući mreži da prepozna složene obrasce i odnose u podacima.

Ovako definirane funkcije predstavljaju jedan sloj rekurentne neuronske mreže. U nastavku rada u kontekstu biblioteke keras koristit će se RNN s jednim rekurentnim slojem. Slika 2.1 prikazuje shemu rekurentne neuronske mreže.




Slika 2.1: Dijagram rekurentne neuronske mreže

## 2.2 *Izazovi i ograničenja*

Iako rekurentne neuronske mreže imaju izvrsne sposobnosti u radu sa sekvencijalnim podacima, postoje određena ograničenja koja treba uzeti u obzir. Moderna literatura navodi sljedeće izazove [17]:

1. **Vrijeme treniranja i računska složenost**

   Rekurentne neuronske mreže, posebno one s višestrukim rekurentnim slojevima, mogu biti vrlo zahtjevne u pogledu vremenske i računske složenosti. Zbog sekvencijalne prirode obrade podataka, njihova paralelizacija je ograničena, što produžuje vrijeme treniranja, osobito za duže ulazne sekvence. Treniranje složenih RNN arhitektura često zahtijeva specijaliziranu računalnu opremu.

2. **Prenaučenost i generalizacija** (eng. *Overfitting and generalization*)

   Kao i kod drugih modela strojnog učenja, rekurentne neuronske mreže mogu biti podložne prenaučenosti, osobito kada se treniraju na malim skupovima podataka. Prenaučenost nastaje kada model previše zapamti specifične karakteristike skupa za treniranje i ne uspije generalizirati na nove, neviđene podatke. Primjena tehnika regularizacije i pažljiv izbor parametara, poput težina i vektora pristranosti, može pomoći u ublažavanju ovog problema.

3. **Nedostatak interpretabilnosti**

   Rekurentne neuronske mreže često se nazivaju "crnim kutijama" jer je njihovo unutarnje funkcioniranje i proces donošenja odluka teško razumjeti i interpretirati. Postoji stalno rastuće područje istraživanja koje se bavi razvojem inter-

pretabilnih arhitektura i tehnika za objašnjavanje predikcija koje RNN modeli daju [9].

4. **Rukovanje dugoročnim ovisnostima**

   Iako su RNN-ovi dobri u prepoznavanju kratkoročnih i srednje dugoročnih ovisnosti, suočavaju se s izazovima u prepoznavanju dugoročnih ovisnosti [15], posebno kada između dva važna podatka postoji velika udaljenost. Arhitekture poput Long Short-Term Memory (LSTM) i Gated Recurrent Unit (GRU) razvijene su kako bi barem djelomično riješile ove poteškoće. Više o LSTM mrežama bit će objašnjeno u kasnijim poglavljima.

5. **Skaliranje i izazovi s paralelizacijom**

   Skaliranje rekurentne neuronske mreže kako bi radila s velikim skupovima podataka ili kompleksnim zadacima je izazovno zbog memorije koju ovakvi zadaci iziskuju. Kako bi se riješio taj problem koriste se tehnike procesiranja malih serija podataka. Paralelizacija predstavlja izazov jer je u radu sa sekvencijalnim podacima jako bitan red procesiranja podataka i uvođenje nekog oblika konkurentnosti je otežano.

Rad na minimiziranju ovih izazova ključ je za daljnji razvoj rekurentnih neuronskih mreža.

## *2.3 Treniranje i poteškoće pri treniranju*

Proces treniranja odvija se u tri koraka, to su unaprijedni prolaz, unazadni prolaz kroz vrijeme i ažuriranje težina. Pri unaprijednom prolazu iz ulaznog vektora $x^{(t)}$ i skrivenog stanja prethodnog koraka $h^{(t-1)}$ računa se predviđanje $\hat{y}^{(t)}$;

$$\hat{y}^{(t)} = f(x^{(t)}, h^{(t-1)}) \tag{2.6}$$

Za svaki vremenski korak računa se gubitak te na kraju ukupni gubitak.

Po završetku unaprijednog prolaza slijedi unazadna propagacija kroz vrijeme. Tijekom unazadne propagacije kroz vrijeme, računa se gradijent gubitka u odnosu na parametre modela, $W_{hx}, W_{hh}, W_{yh}, b_h, b_y$, prema pripadajućim jednadžbama. Izračunati gradijenti koriste se kako bi se ažurirale težine korištenjem optimizacijskih

algoritama. Primjer takvih algoritama je Stochastic Gradient Descent (SGD) kojeg prvi uvode Robbinson i Monro [18] 1951. godine. Koriste se i ostale varijante SGD-a Adam ili RMSprop. Nadalje, za što veću učinkovitost pri treniranju rekurentne mreže potrebno je dobro odabrati početne vrijednosti težina. Loša inicijalizacija težina može dovesti do saturacije aktivacijske funkcije ili gradijenta [19], što može usporiti proces učenja. Inicijalizacijske strategije kao Xavier [20] ili HE inicijalizacija najčešće se koriste kako bi se osigurala što bolja inicijalizacija težina.

U unaprijednom prolazu, skriveno se stanje $h^{(t)}$ postavlja na nula ili na nasumično izabranu vrijednost. Zatim se za svaki vremenski korak t računa $h^{(t)}$ prema jednadžbi (2.2). Izlazna vrijednost računa se na temelju vrijednosti skrivene funkcije $h^{(t)}$ prema jednadžbi (2.4). Tako izračunati $\hat{y}^{(t)}$ uspoređuje se sa stvarnom izlaznom vrijednošću te se računa gubitak na temelju odabrane funkcije gubitka. Funkcija gubitka općenito se definira kao:

$$L : (y, \hat{y}) \rightarrow \mathbb{R}_{\geq 0} \tag{2.7}$$

Primjer funkcije gubitka koja se često koristi je srednji kvadrat greške (eng. *Mean squared error*).

$$L(y^{(1)}, ..., y^{(t_0)}, \hat{y}^{(1)}, ..., \hat{y}^{(t_0)}) = \frac{1}{t_0} \sum_{t=1}^{t_0} \sqrt{\frac{1}{m} \sum_{i=1}^{m} (y_i^{(t)} - \hat{y}_i^{(t)})^2} \tag{2.8}$$

Jednadžba (2.8) može poprimiti i druge oblike, ovisno koja funkcija gubitka je odabrana. U kasnijoj implementaciji koda bit će definirana i korištena poprečna entropija kao funkcija gubitka.

Programski se srednji kvadrat greške može ostvariti kao:

```
def MSE(target, y_hat):
    squaredError = (target - y_hat)**2
    sumSquaredError = numpy.sum(squaredError)
    mse = sumSquaredError / y_hat.size
    return mse
```

Zatim slijedi unazadna propagacija kroz vrijeme gdje se računa gradijent gubitka u odnosu na vrijednosti težina $W_{hx}$, $W_{hh}$, $W_{yh}$ i vektore pristranosti $b_h$ i $b_y$ [4].

$$\begin{aligned}\frac{\partial L}{\partial W_{hx}} &= \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_{hx}} = \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_{hx}} \\ &= \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot \frac{\partial}{\partial W_{hx}} \sigma(W_{hx} \cdot x^{(t)} + W_{hh} \cdot h^{(t-1)} + b_h) \\ &= \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot x^{(t)}\end{aligned} \tag{2.9}$$

$$\begin{aligned}\frac{\partial L}{\partial W_{hh}} &= \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_{hh}} = \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_{hh}} \\ &= \sum_t \frac{\partial L}{\partial h^{(t)}} \cdot \frac{\partial}{\partial W_{hh}} \sigma(W_{hx} \cdot x^{(t)} + W_{hh} \cdot h^{(t-1)} + b_h) \\ &= \sum_t h^{(t-1)} \cdot \frac{\partial L}{\partial h^{(t)}}\end{aligned} \tag{2.10}$$

$$\frac{\partial L}{\partial W_{yh}} = \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}}{\partial W_{yh}} = \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial}{\partial W_{yh}} (W_{yh} h^{(t)} + b_y) = \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot h^{(t)} \tag{2.11}$$

$$\frac{\partial L}{\partial b_h} = \sum_t \frac{\partial L}{\partial h^{(t)}} \tag{2.12}$$

$$\frac{\partial L}{\partial b_y} = \sum_t \frac{\partial L}{\partial \hat{y}^{(t)}} \tag{2.13}$$

$$\frac{\partial L}{\partial h^{(t)}} = \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} = \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot \frac{\partial}{\partial h^{(t)}} (W_{yh} h^{(t)} + b_y) = \frac{\partial L}{\partial \hat{y}^{(t)}} \cdot W_{yh} \tag{2.14}$$

Ove jednadžbe predstavljaju shematski opis unazadne propagacije kroz vrijeme te kao takve opisuju veze između varijabli, suma i gradijenata na apstraktan način. One ne predstavlja konkretne korake koji se mogu implementirati i prevesti u kod. Jednadžbe definiraju što treba biti izračunato, ali ne specificiraju kako na optimalan način proći kroz sve vremenske korake te kako točno koristiti strukture podataka kao što su 2D polja i tenzori, za to je potrebno koristiti prilagodbe. Tako u jednadžbi (2.11) za ostvarivanje koda će se morati koristiti transponirana matrica skrivenog stanja $h^{(t)T}$ kako bi umjesto vektora stupca dimenzija (H, 1) dobili vektor dimenzija (1, H) i time omogućili matrično množenje s $\frac{\partial L}{\partial \hat{y}^{(t)}}$ koje ima dimenzije (H, 1). Razlog

za transponiranje $W_{yh}$ u jednadžbi (2.14), $x^{(t)}$ u jednadžbi (2.9) i $h^{(t-1)}$ u jednadžbi (2.10) ekvivalentan je.

Posljednji korak treninga mreže završava ažuriranjem težina sukladno rezultatima dobivenim u prethodnim koracima. Općenito pravilo po kojem se ažuriraju težine uz upotrebu SGD algoritma za optimizaciju [18] je:

$$\Delta W = -\texttt{lr} \cdot \frac{\partial L}{\partial W} \tag{2.15}$$

lr predstavlja stopu učenja (eng. *learning rate*), jedan od tzv. hiperparametara mreže - vrijednost koja se proizvoljno zadaje na početku treninga. Iz jednadžbe je vidljivo da veća stopa učenja rezultira većim izmjenama vrijednosti težina što može rezultirati divergencijom ili oscilacijama zbog kojih model nikada ne postigne željeni minimum. Nasuprot tome, mala stopa učenja dovodi do manjih izmjena vrijednosti težina, ali usporava proces obuke jer je potrebno mnogo epoha da se postigne konvergencija u lokalni ili globalni minimum [4].

Kao što je ranije spomenuto jedan od problema pri treniranju jest da može doći do prenaučenosti. U sprječavanju prenaučenosti i kako bi se poboljšale generalizacijske sposobnosti modela koriste se tehnike reguliranja poput metode regulacija L1 [22] i L2 [23], dropout i batch normalizacija [4].

Treniranje rekurentnih mreža može brzo postati zahtjevno za računala te se iz tog razloga koristi treniranje u serijama (eng. *Batch training*) i mini-serijsko procesiranje (eng. *Mini-batch processing*) kako bi se poboljšala računska efikasnost. U kombinaciji s tim metodama može se koristiti i SGD. Umjesto ažuriranja težina nakon procesiranja svakog pojedinog uzorka, model prvo procesira seriju uzoraka, a težine su ažurirane na temelju akumuliranog gradijenta te serije. Mini-serijsko procesiranje dopušta paralelno računanje i bržu konvergenciju i time omogućava treniranje rekurentnih mreža na većim serijama podataka.

Rad s različitim duljinama ulaznih podataka još je jedan od izazova treniranja rekurentnih mreža. Dodavanje razmaka ili drugih znakova podatcima (eng. *padding*) kako bi imali konstantnu duljinu je čest način da se izbjegne ova poteškoća, ali to opet dovodi do izvođenja dodatnih računskih koraka te može dovesti do neefektivnog korištenja memorije. Kako bi se minimizirao padding i optimizirao proces treniranja koriste se tehnike sortiranja sekvenci po duljini ili grupiranje [21].

## 2.4 Implementacija jednostavne rekurentne neuronske mreže

Pogledajmo kako bi opisanu rekurentnu neuronsku mrežu implementirali u Pythonu koristeći samo biblioteku numpy [24]. Cijeli kod dostupan je i na [25] te ga je moguće isprobati na [26]. Dio koda preuzeto je s [27]. Za aktivacijsku funkciju koristi se tangens hiperbolni. Gubitak će se računati prema srednjem kvadratu greške te će sve težine i pristranosti biti zadane nasumično. Skup podataka za treniranje sastoji se od niza vrijednosti generiranih iz jednog nasumično odabranog broja između 1 i 100. Taj broj prolazi kroz tri računske operacije: prvo se množi s 10, zatim se kvadrira, a na kraju dijeli s 2. Na primjer, ako je nasumično generiran broj 2, pripadajući skup podataka bit će [20, 4, 1]. Zadatak mreže je na temelju polja brojeva koje je dobila naučiti koji je početni nasumično generirani broj.

```
import numpy as np


class SimpleRNN:
    def __init__(self, input_size, hidden_size, output_size,lr):
        # Inicijalizacija težina
        self.Wxh = np.random.randn(hidden_size, input_size) * 0.01
        self.Whh = np.random.randn(hidden_size, hidden_size) * 0.01
        self.Wyh = np.random.randn(output_size, hidden_size) * 0.01
        self.bh = np.zeros((hidden_size, 1))
        self.by = np.zeros((output_size, 1))
        self.lr =lr
    # Unaprijedni prolaz
    def forward(self, inputs, h_prev):
        self.h_prev = h_prev
        self.hs = {}
        self.hs[-1] = h_prev
        self.outputs = []
        for t in range(len(inputs)):
            self.hs[t] = np.tanh(np.dot(
                self.Wxh, inputs[t]) + np.dot(self.Whh, self.hs[t-1])
                + self.bh)
```

```
            y = np.dot(self.Wyh, self.hs[t]) + self.by
            self.outputs.append(y)
        return self.outputs, self.hs[len(inputs) - 1]
    # Unazadni prolaz
    def backward(self, inputs, targets, h_prev, lr=0.01):
        dWxh, dWhh, dWyh = np.zeros_like(self.Wxh),
            np.zeros_like(self.Whh), np.zeros_like(self.Wyh)
        dbh, dby = np.zeros_like(self.bh), np.zeros_like(self.by)
        dh_next = np.zeros_like(h_prev)
        # BPTT
        loss = 0
        for t in reversed(range(len(inputs))):
            dy = self.outputs[t] - targets[t]
            loss += 0.5 * np.sum(dy ** 2)
            dWyh += np.dot(dy, self.hs[t].T)
            dby += dy
            dh = np.dot(self.Wyh.T, dy) + dh_next
            dh_raw = (1 - self.hs[t] ** 2) * dh
            dWxh += np.dot(dh_raw, inputs[t].T)
            dWhh += np.dot(dh_raw, self.hs[t - 1].T)
            dbh += dh_raw
            dh_next = np.dot(self.Whh.T, dh_raw)
        # Gradient clipping
        for dparam in [dWxh, dWhh, dWyh, dbh, dby]:
            np.clip(dparam, -1, 1, out=dparam)
        # Ažuriranje težina i pristranosti
        self.Wxh -= lr * dWxh
        self.Whh -= lr * dWhh
        self.Wyh -= lr * dWyh
        self.bh -= lr * dbh
        self.by -= lr * dby
        return loss
    # Trening
```

```
    def train(self, inputs, targets, h_prev):
        outputs, h_prev = self.forward(inputs, h_prev)
        loss = self.backward(inputs, targets, h_prev)
        return outputs, h_prev, loss

# Generiranje nasumičnog broja i polja s podacima za trening
def generate_custom_sequence_data(start=None):
    if start is None:
        start = np.random.randint(1, 100)
    inputs = np.array([
        [[start*10]],
        [[start ** 2]],
        [[start / 2]]
    ])
    targets = np.array([
        [[start]],
        [[start]],
        [[start]]
    ])
    return inputs, targets

# Stvaranje objekta klase SimpleRNN
rnn = SimpleRNN(input_size=1, hidden_size=5, output_size=1,
    lr=0.001)
h_prev = np.zeros((5, 1))

# Trening
for epoch in range(100000):
    inputs, targets = generate_custom_sequence_data()
    outputs, h_prev, loss = rnn.train(inputs, targets, h_prev)

    if epoch % 100 == 0:
        print(f'Epoch {epoch} outputs:')
```

```
        for t, output in enumerate(outputs):
            print(f'Time step {t}: input {inputs[t].ravel()[0]:.4f},
            predicted {output.ravel()[0]:.4f},
            target {targets[t].ravel()[0]:.4f}')
            print(f'Wxh:\n{rnn.Wxh}')
            print(f'Whh:\n{rnn.Whh}')
            print(f'Wyh:\n{rnn.Wyh}')
            print(f'bh:\n{rnn.bh}')
            print(f'by:\n{rnn.by}')
            print(f'Loss: {loss}\n')
```

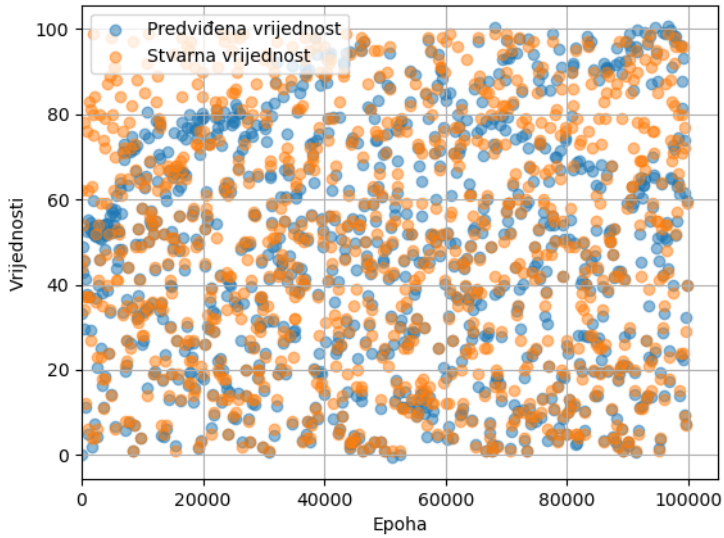
SimpleRNN unutar biblioteke odgovara formulama od (2.1) do (2.4) pri čemu je $H = p$, $b_y = 0$, a $W_{yh}$ jednak je jediničnoj matrici. Ispis vrijednosti parametara i rezultata za 0., 500000. i 1000000. epohu moguće je vidjeti u dodatku A. Dobivena predviđanja nakon treninga mreže 1000000 epoha na brojevima između 1 i 100 blizu su stvarnim rezultatima. Pogledajmo za primjer jedan ispis:

```
Time step 0: input 70.0000, predicted 10.9760, target 7.0000
Time step 1: input 49.0000, predicted 6.7758, target 7.0000
Time step 2: input 3.5000, predicted 7.6257, target 7.0000
```

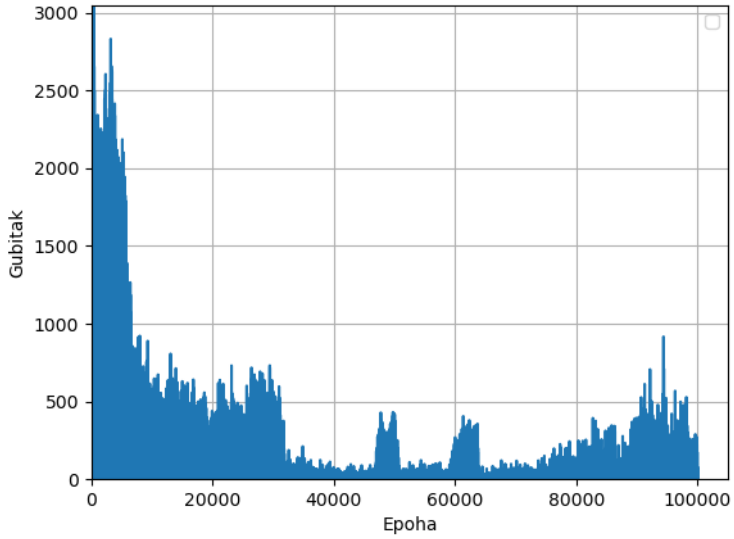
U većini primjera se pokazalo da će se najbolje predviđanje dati za primjere u kojima je ulazni broj iznosom najbliži ciljanom broju, bez obzira na tip računske operacije kojom se došlo od ciljanog broja do ulaznog broja. Na grafu 2.3 prikazana su najbolja od tri predviđanja. Može se uočiti kako su odstupanja od ciljanih vrijednosti veća u početku te su veća što je broj koji je potrebno predvidjeti veći.

Pokazalo se kako povećanje broj skrivenih stanja i epoha daje bolje rezultate. Brzina učenja može se povećati sve do 0.1 te je tada moguće smanjiti broj epoha na 100000 i dobivati slične rezultate. Za brzine učenja veće od jedan predviđanja počnu težiti u konstantnu vrijednost.

U nastavku ovog rada mreže ćemo trenirati na tekstovima koje predstavljaju izazov zbog dugoročnih ovisnosti koje je potrebno uočiti kako bi se mogao generirati smisleni tekst, trening će se provoditi u serijama te će se parametri modela ažurirati svakih nekoliko serija iz gradijenta funkcije gubitka koji se akumulirao tijekom njih.




Slika 2.2: Graf predviđenih i ciljanih vrijednosti za brojeve od 1 do 100




Slika 2.3: Graf gubitka

Slova i znakove pretvarat ćemo u brojčane vrijednosti te će račun teći kao i u gornjem primjeru.

Pogledajmo pobliže što bi bilo kada bismo htjeli ovakvu mrežu trenirati na tekstu. Promotrimo rečenicu: "Mačka je sjela na tepih.". Odnos između "mačka" i "sjela" ključan je za razumijevanje značenja rečenice. Međutim, taj odnos može obuhvatiti nekoliko riječi ili vremenskih koraka. Uočavanje te ovisnosti predstavlja izazov za mreže. Gradijenti igraju presudnu ulogu u rješavanju ovog izazova s obzirom na to da daju mjerilo koliko bi se parametri mreže trebali promijeniti kako bi izlazni

rezultati bili što bliži očekivanima te tako direktno utječu na rezultate od prvog do zadnjeg koraka treninga. Kada se mreža razvija kroz više vremenskih koraka tijekom unazadne propagacije kroz vrijeme, gradijenti se računaju za svaki korak, ali rezultati prethodnih koraka utječu na svaki idući korak, omogućujući modelu da nauči ovisnosti koje obuhvaćaju značajan broj vremenskih koraka. Posebno, nestajanje ili nagli rast gradijenta može dovesti do nestabilnosti i omesti sposobnost modela za uočavanje dugoročnih ovisnosti. U sljedećem poglavlju pobliže ćemo proučiti gradijent.

## *2.5 Gradijent*

Gradijenti su u srcu obuke neuronskih mreža. Oni označavaju smjer i veličinu promjene parametara modela sukladno rezultatima funkcije gubitka. Ti parametri podrazumijevaju ranije spomenute težine i pristranosti. U kontekstu rekurentnih neuronskih mreža, važnost gradijenta proteže se dalje od samog ažuriranja težina – gradijenti mreži omogućavaju da nauči vremenske obrasce i odnose unutar sekvencijalnih podataka. U rekurentnoj mreži se gradijenti izračunavaju koristeći algoritam unazadne propagacije kroz vrijeme na način kako je opisano u prijašnjem poglavlju. Učinkovito računanje gradijenata prilikom unazadne propagacije nije važno samo za uočavanje dugoročnih ovisnosti, već i za ubrzanje procesa obuke. Informacije o gradijentima usmjeravaju ažuriranja parametara u pravom smjeru i omogućuju modelu brže konvergiranje prema rješenju. Osim brzine obuke, gradijenti igraju ulogu u generalizaciji. Model koji ispravno nauči parametre moći će dobro generalizirati na neviđenim podacima, budući da je učinkovito naučio temeljne obrasce i odnose unutar sekvencijalnih nizova na kojim je proveden trening.

Neki od problema gradijenta s kojima se susrećemo kod rekurentnih neuronskih mreža jesu problem nestajućih gradijenata i problemom eksplodirajućih gradijenata [4]. Ovi problemi proizlaze iz prirode unazadne propagacije i množenja gradijenata kako se šire kroz vrijeme. Do problema nestajućih gradijenata dolazi kada se gradijenti u više iteracija iznova množe s težinama manjim od jedan. Ovo uzrokuje eksponencijalno smanjivanje gradijenta kako se širi unatrag kroz vrijeme. Ovaj fenomen ometa sposobnost mreže da nauči dugoročne ovisnosti jer gradijenti praktički nestaju, sprječavajući značajna ažuriranja parametara modela [28]. Suprotno tome, gradijenti također mogu eksplodirati kada se ponavljaju množenjem

s težinama većim od 1. To dovodi do izuzetno velikih ažuriranja parametara modela što može destabilizirati proces obuke. Oba problema mogu ometati sposobnost mreža da učinkovito uoče vremenske uzorke u nizovima.

Pogledajmo jednostavni primjer, kao što je navedeno u [4], rekurencije bez nelinearne aktivacijske funkcije i bez ulaza x

$$h^{(t)} = W^T h^{(t-1)} \tag{2.16}$$

gdje je $h^{(t)} \in \mathbb{R}^H$ skriveno stanje, $W^T \in \mathbb{R}^{H \times H}$ transponirana težinska matrica. Pretpostavlja se da je ta matrica i simetrična. Ovaj izraz može se pojednostaviti na oblik:

$$h^{(t)} = (W^t)^T h^{(0)} \tag{2.17}$$

Težina W ima vlastitu dekompoziciju oblika

$$W = Q \Lambda Q^T \tag{2.18}$$

gdje je $Q \in \mathbb{R}^{H \times H}$ ortogonalna matrica vlastitih vektora od W, $\Lambda \in \mathbb{R}^{H \times H}$ je dijagonalna matrica vlastitih vrijednosti od W. S obzirom na to da je Q ortogonalna vrijedi:

$$Q^T Q = I \longrightarrow Q^T = Q^{-1} \tag{2.19}$$

gdje je I matrica identiteta. Početni izraz sada ima oblik:

$$h^{(t)} = Q^T \Lambda^t Q h^{(0)} \tag{2.20}$$

iz toga slijedi da

$$\text{ako } |\Lambda_{ii}| < 1, \Lambda_{ii}^t \to 0 \tag{2.21}$$

$$\text{ako } |\Lambda_{ii}| > 1, \Lambda_{ii}^t \to \infty \tag{2.22}$$

Ovisno o iznosu vlastite vrijednosti $\Lambda$ cijeli rekurentni izraz može $h^{(t)}$ težiti u nulu i gradijent će iščezavati ili izraz može težiti u beskonačnost te će gradijent eksplodirati. U praksi, češće se javlja problem nestajućeg gradijenta. Srećom, razvijene su različite tehnike za ublažavanje ovih problema. Neke od najpoznatijih tehnika su tehnike inicijalizacije težina, ograničavanja gradijenta, tehnika arhitekture s vratima i tehnika

preskočne veze [28]. Opišimo neke od spomenutih tehnika.

1. **Tehnike inicijalizacije težina**

   Pravilnim postavljanjem početnih težina moguće je kontrolirati opseg veličina gradijenata, sprječavajući tako da njihove vrijednosti postanu prevelike ili premale. Već je spomenuto kako se početne težine mogu inicijalizirati nasumično, ali kako bi se osigurali što bolji rezultati razvijene su tehnike inicijalizacije težina, poput Xavierove uniformne inicijalizacije. Kod Xavierove uniformne inicijalizacije [20] potrebno je izabrati težine W iz nasumične uniformne distribucije u rasponu od $-x$ do $x$ gdje je $x$:

   $$x = \sqrt{\frac{6}{n+p}} \tag{2.23}$$

   gdje $n$ označava veličinu ulaznog vektora, a $p$ veličinu izlaznog vektora.

2. **Ograničavanje gradijenta**

   Ograničavanje gradijenata podrazumijeva postavljanje gornje granice za gradijente tijekom unazadne propagacije. To sprječava problem eksplodirajućih gradijenata osiguravajući da veličine gradijenata ostanu unutar razumne granice.

3. **Tehnike arhitekture s vratima**

   Arhitekture poput Long Short-Term Memory (LSTM) [29] i Gated Recurrent Unit (GRU) [34] vrste su Arhitekture s vratima (eng. *Gated Architectures*) i specifično su osmišljene kako bi se riješio problem nestajućih gradijenata. One uključuju mehanizme s vratima koji omogućuju mreži da kontrolira tok informacija i gradijente, omogućujući im učinkovitije uočavanje dugoročnih ovisnosti. U sljedećem poglavlju bit će više rečeno o LSTM mrežama.

# 3 Long short term memory

Long short-term memory vrsta je rekurentne neuronske mreže dizajnirane kako bi se pokušao riješiti problem nestajućeg i eksplodirajućeg gradijenta koji se javlja pri treniranju rekurentne mreže. Prvi puta se spominju 1997. u radu S. Hochreiter i J. Schmidhuber [29]. Za razliku od rekurentnih mreža, LSTM mreže mogu zapamtiti informacije tijekom dužih vremenskih perioda što ih čini jako korisnim u procesiranju sekvencijalnih podataka s ovisnostima koje je moguće uočiti tek nakon analize velikog broja podataka. Kao i rekurentne neuronske mreže, LSTM mreže danas se koriste u različitim područjima kao što su procesiranje prirodnog jezika, prepoznavanje govora i analiza vremenskih serija podataka [30].

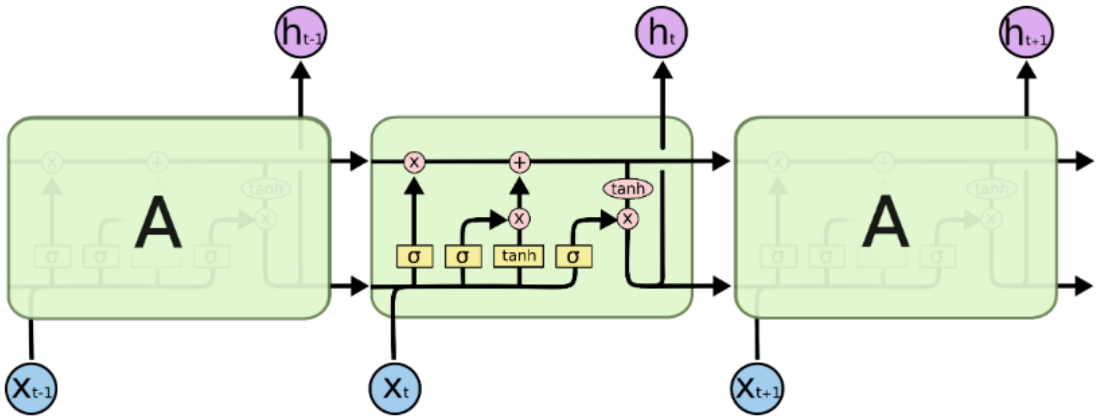
## *3.1 LSTM arhitektura i način rada*

LSTM, uz komponente koje sadrže RNN, sadrže i vrata koja kontroliraju što se pamti ili zaboravlja pri svakom koraku u sekvenci. Postoje vrata za zaborav (eng. *forget gate*) koja odbacuju irelevantne informacije iz stanja ćelije, vrata za unos (eng. *input gate*) koja dodaju nove relevantne informacije te vrata za izlaz (eng. *output gate*) koja odlučuju koje će se informacije koristiti za predviđanja. Ovaj mehanizam omogućava modelu da pamti važne pojedinosti tijekom dugih sekvenci, čineći ga učinkovitim u zadacima koji uključuju sekvence poput modeliranja jezika [30].

Operacije unutar LSTM mreže odvijaju se u vremenskim koracima. Za svaki vremenski korak t, odvijaju se operacije ulaznog računa, ažuriranja stanja ćelije, ažuriranje skrivenog stanja te izlazni račun.

Stanje ćelije $C^{(t)}$ predstavlja dugoročnu memoriju LSTM mreže dok skriveno stanje $h^{(t)}$ čuva kratkoročne ovisnosti i prenosi ih idućem vremenskom koraku. Vrijednosti izračunate na forget i input gate ulaze u ovo stanje ćelije, koje čuva i prenosi relevantne informacije kroz vrijeme. Za razliku od drugih dijelova mreže, stanje ćelije nije izravno izračunato iz težina i vektora pristranosti, što pomaže u sprječavanju problema poput eksplodiranja ili nestanka gradijenta. Stanje ćelije omogućuje LSTM mreži da selektivno pamti i zaboravlja informacije na temelju podataka koje dobiva s različitih vrata. Ovo je ključna komponenta LSTM mreže jer omogućuje zadržavanje informacija tijekom duljih vremenskih intervala.

Sljedeće jednadžbe dane su u [4] i [35]. Prvi korak rada LSTM mreže odvija se




Slika 3.1: Skica građevne ćelije long short-term memory mreže (izvor: https://colah.github.io/posts/2015-08-Understanding-LSTMs/, preuzeto 10.11.2023.)

na forget gate-u. Forget gate za svaku vremensku jedinicu t uzima ulazne podatke $x^{(t)} \in \mathbb{R}^n$, prethodno skriveno stanje $h^{(t-1)} \in [-1,1]^p$ te daje izlazne podatke $f^{(t)} \in [0,1]^p$

$$f^{(t)} = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f) \tag{3.1}$$

gdje je izlazna težina $W_f \in \mathbb{R}^{p \times p}$, ulazna težina $U_f \in \mathbb{R}^{p \times n}$ i pristranost $b_f \in \mathbb{R}^p$. Rezultat dobiven na forget gateu određivat će koliko će informacija iz prethodnog stanja ćelije biti zaboravljeno. Što je taj rezultat bliži nuli to će podaci imati manji utjecaj na buduće rezultate.

U drugom dijelu nalaze se ulazna vrata. Ulazna vrata za svaku vremensku jedinicu t uzimaju ulazne podatke $x^{(t)} \in \mathbb{R}^n$, prethodno skriveno stanje $h^{(t-1)} \in [-1,1]^p$ te daje izlazne podatke $i^{(t)} \in [0,1]^p$.

$$i^{(t)} = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i) \tag{3.2}$$

gdje je izlazna težina $W_i \in \mathbb{R}^{p \times p}$, ulazna težina $U_i \in \mathbb{R}^{p \times n}$ i pristranost $b_i \in \mathbb{R}^p$.

U trećem bloku, na izlaznim vratima, za svaku vremensku jedinicu t uzimaju se ulazni podaci $x^{(t)} \in \mathbb{R}^n$, prethodno skriveno stanje $h^{(t-1)} \in [-1,1]^p$ te se računa $o^{(t)} \in [0,1]^p$.

$$o^{(t)} = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o) \tag{3.3}$$

gdje je izlazna težina $W_o \in \mathbb{R}^{p \times p}$, ulazna težina $U_o \in \mathbb{R}^{p \times n}$ i pristranost $b_o \in \mathbb{R}^p$.

Kako bi se stanje ćelije moglo konačno ažurirati potrebno je još izračunati i novo stanje ćelije, $\tilde{c}^{(t)} \in [-1, 1]^p$

$$\tilde{c}^{(t)} = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c) \tag{3.4}$$

gdje je izlazna težina $W_c \in \mathbb{R}^{p \times p}$, ulazna težina $U_c \in \mathbb{R}^{p \times n}$ i pristranost $b_c \in \mathbb{R}^p$.

U posljednjem koraku računa se nova vrijednost stanja ćelije s pomoću Hadamardovog produkta [14] dva vektora.

Definicija Hadamardovog produkta: neka je $x \in \mathbb{R}^n$ i $y \in \mathbb{R}^n$ tada

$$x \odot y = (x_1 y_1, x_2 y_2, ..., x_n y_n) \tag{3.5}$$

Vrijednost stanja ćelije definira se kao

$$C^{(t)} = (f^{(t)} \odot C^{(t-1)}) + (i^{(t)} \odot \tilde{c}^{(t)}) \tag{3.6}$$

gdje je $C^{(t-1)} \in \mathbb{R}^p$ stanje ćelije prijašnjeg vremenskog koraka.

Nakon ažuriranja stanja ćelije još je potrebno odrediti izlazne podatke. Za to je potrebno izračunati skriveno stanje $h^{(t)} \in [-1, 1]^p$

$$h^{(t)} = o^{(t)} \odot \tanh(C^{(t)}) \tag{3.7}$$

Konačni izlazni podaci $y^{(t)} \in \mathbb{R}^p$ definirani su kao:

$$y^{(t)} = W_y h^{(t)} + b_y \tag{3.8}$$

gdje je $W_y \in \mathbb{R}^{q \times p}$ matrica težine i $b_y \in \mathbb{R}^q$ pristranost.

## *3.2 Nedostaci LSTM mreža*

Iako su dizajnirane kako bi se riješili nedostaci koje se javljaju kod RNN, LSTM ima neke vlastite nedostatke zbog kojih daje lošije rezultate u usporedbi s trenutačno najpopularnijim transformativnim modelima.

Poteškoće koje se mogu javiti kod LSTM mreža odnose se na problem pamćenja važnih informacija tijekom dugih sekvenci podataka. Primjer iz literature ( [31], Fi-

gure 2) prikazuje traženje najbližeg susjeda, gdje se za broj 5 kao najbliži susjedi mogu smatrati brojevi 4 i 6. LSTM mreža može ispravno prepoznati broj 4 kao najbližeg susjeda zbog svoje sposobnosti prepoznavanja sekvencijalnih obrazaca, ali ako mreža nije dobro istrenirana ili ako postoji previše podataka koji stvaraju "šumove", mogla bi imati poteškoća s pravilnim pohranjivanjem i prepoznavanjem broja 6 kao drugog najbližeg susjeda.

Druga poteškoća koja se ističe su limitirani kapaciteti za pohranu dugoročnih ovisnosti. Iako LSTM mreže značajno poboljšavaju sposobnost pamćenja prošlih događaja u usporedbi s klasičnim RNN, njihov kapacitet je i dalje ograničen kad se radi o vrlo dugim sekvencama podataka. Dok LSTM može učinkovito povezati riječi unutar rečenice ili čak ovisnosti između nekoliko susjednih rečenica, kada očekujemo da obradi veći broj rečenica, LSTM mreže često imaju poteškoća u prepoznavanju takvih dugoročnih ovisnosti zbog ograničenog kapaciteta za pamćenje.

Također, ističe se i nedostatak paralelizabilnosti. LSTM mreže zahtijevaju sekvencijalno, sinkronizirano procesiranje podataka, što je od iznimne važnosti za pravilan rad i treniranje. To znači da LSTM mreže moraju obrađivati sekvence korak po korak, gdje svaki korak ovisi o prethodnom. Ova sekvencijalna priroda treniranja onemogućava uvođenje konkurentnosti ili asinkronosti u proces treniranja, što ozbiljno ograničava mogućnosti za ubrzavanje treninga. Zbog toga su LSTM mreže sporije u usporedbi s arhitekturama koje podržavaju paralelno procesiranje podataka, poput transformera.

## *3.3 Proširena LSTM (xLSTM)*

Kao pokušaj da se uklone ili minimiziraju svi nedostaci LSTM spomenuti u prijašnjem poglavlju predložen je xLSTM (eng. *Extended LSTM*). Ključni princip koji uvodi xLSTM je eksponencijalno usmjeravanje (eng. *Exponential Gating*) te dvije nove memorijske strukture, sLSTM i mLSTM kao dva nova tipa memorijskih ćelija unutar arhitekture mreže.

sLSTM uvodi novo stanje – stanje normalizatora (eng. *normalizer state*), $n^{(t)} \in \mathbb{R}$, koje se definira kao:

$$n^{(t)} = f^{(t)} n^{(t-1)} + i^{(t)} \tag{3.9}$$

i stanje stabilizatora (eng. *stabilizer state*), $m^{(t)} \in \mathbb{R}^p$:

$$m^{(t)} = max(log(f^{(t)}) + m^{(t-1)}, log(i^{(t)})) \tag{3.10}$$

gdje $i^{(t)}$ koristi eksponencijalnu funkciju umjesto sigmoidne funkcije kao u (3.2). Novo skriveno stanje definira se kao:

$$h^{(t)} = o^{(t)} \frac{C^{(t)}}{n^{(t)}} \tag{3.11}$$

Stabilizirano stanje sadrži vlastita input i forget vrata.

$$f^{(t)} \rightarrow f^{(t)s} = f^{(t)} exp(m^{(t-1)} - m^{(t)}) \tag{3.12}$$

$$i^{(t)} \rightarrow i^{(t)s} = i^{(t)} exp(-m^{(t)}) \tag{3.13}$$

gdje su $f^{(t)}$ i $i^{(t)}$ zadani kao u pripadajućim jednadžbama (3.1) i (3.2).

Stanje stabilizirane ćelije i normalizirano stanje mogu se i redefinirati kao:

$$C^{(t)} \rightarrow C^{(t)s} = C^{(t)} exp(-m^{(t)}) \tag{3.14}$$

$$n^{(t)} \rightarrow n^{(t)s} = n^{(t)} exp(-m^{(t)}) \tag{3.15}$$

Za razliku od LSTM koja koristi vektorsku memoriju, sLSTM koristi skalarnu memoriju. Ovo ograničenje omogućuje da mreža radi s manjim brojem parametara te je trening brži zbog smanjenog broj računskih operacija. Nedostatak skalarne memorije je ograničen kapacitet za pohranu informacija, odnosno teško je pohranjivati informacije iz prijašnjih vremenskih koraka i tako je smanjena sposobnost uočavanja dugoročnih ovisnosti. Kako bi se to ograničenje prevladalo koristi se miksanje memorije (eng. *memory mixing*) uz naglasak na eksponencijalno usmjeravanje. Eksponencijalno usmjeravanje omogućuje veću fleksibilnost protoka informacija, osobito kada je potrebno revidirati odluke pohranjene u memoriji, kao u primjeru s odabirom najbližeg susjeda. U relacijama (3.1) i (3.2) za aktivacijsku funkciju može odabrati eksponencijalna funkcija umjesto sigmoidne. Tako pri donošenju odluka utjecaj pojedinog ulaza se ne mora skalirati između 0 i 1 već njegov utjecaj na buduće rezultate može eksponencijalno varirati.

Memorija mLSTM predstavljena je matricom, što omogućuje pohranjivanje i dohvaćanje kompleksnijih informacija. Također, mLSTM je u potpunosti paralelizabilna

te koristi ažuriranje kovarijance za matričnu memoriju. Ovaj novi tip memorijske ćelije definira novo normalizirano stanje te jednadžbe:
Stanje ćelije, $C^{(t)} \in \mathbb{R}^{p \times p}$

$$C^{(t)} = f^{(t)} C^{(t-1)} + i^{(t)} v^{(t)} k^{(t)T} \tag{3.16}$$

Normalizirano stanje, $n^{(t)} \in \mathbb{R}^p$

$$n^{(t)} = f^{(t)} n^{(t-1)} + i^{(t)} k^{(t)} \tag{3.17}$$

Skriveno stanje, $h^{(t)} \in \mathbb{R}^p$

$$h^{(t)} = o^{(t)} \odot \frac{C^{(t)} q^{(t)}}{max\{|n^{(t)} q^{(t)}|, 1\}} \tag{3.18}$$

Ulazni upit (eng. *query input*), $q^{(t)} \in \mathbb{R}^p$

$$q^{(t)} = W_q x^{(t)} + b_q \tag{3.19}$$

Ulazni ključ (eng. *key input*), $k^{(t)} \in \mathbb{R}^p$

$$k^{(t)} = \frac{1}{\sqrt{p}} W_k x^{(t)} + b_k \tag{3.20}$$

gdje je $p$ dimenzija skrivenog stanja. Ulazne vrijednosti (eng. *value input*), $v^{(t)} \in \mathbb{R}^p$

$$v^{(t)} = W_v x^{(t)} + b_v \tag{3.21}$$

Input gate, $i^{(t)} \in \mathbb{R}$

$$i^{(t)} = exp(w_i x^{(t)} + b_i) \tag{3.22}$$

Forget gate, $f^{(t)} \in \mathbb{R}$,

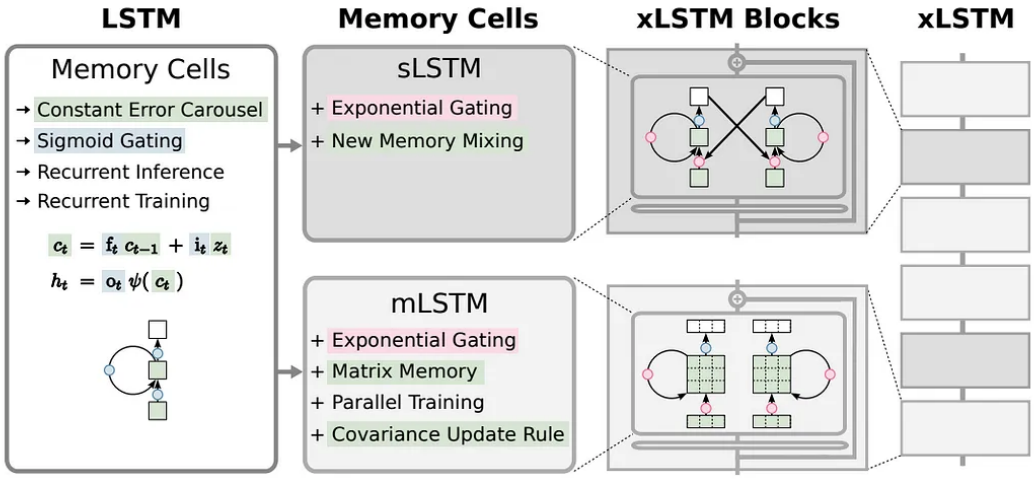
$$f^{(t)} = exp(w_f x^{(t)} + b_f) \tag{3.23}$$

Output gate, $o^{(t)} \in \mathbb{R}^p$

$$o^{(t)} = \sigma(W_o x^{(t)} + b_o) \tag{3.24}$$

gdje su težine $W_k, W_o, W_q, W_k \in \mathbb{R}^{p \times n}$, težinski vektori $w_i, w_f \in \mathbb{R}^n$ i pristranosti $b_k, b_o, b_q, b_k \in \mathbb{R}^p$, $b_i, b_f \in \mathbb{R}$.

Kao što je vidljivo iz jednadžbi stanje ćelije predstavljeno je matricom, ne skalarom te se u računu ne koristi skriveno stanje prethodnog vremenskog koraka. Također, za računanje skrivenog stanja $h^{(t)}$ nije potrebno poznavanje vrijednosti prethodnog skrivenog stanja. S obzirom na to da ne ovise o prethodnom skrivenom stanju moguće je paralelno procesiranje ulaznih vrijednosti. Iako je za izračun novog stanja ćelije potrebna prethodna vrijednost stanja ćelije xLSTM imaju mehanizme poput prefiksa sume kako bi paralelizirale i taj proces.




Slika 3.2: Shema xLSTM mreže (izvor: https://arxiv.org/abs/2405.04517, pristupljeno 28.8.2024)

Na slici 3.2 prikazana je shema xLSTM. Prema testovima provedenim u istraživanju [29] rad xLSTM mreže uspoređen je s radom LSTM, transformativnih mreža i RNN. Zadaci su bili napravljeni kako bi se testirao rad s velikim jezičnim modelima, rad s dugoročnim ovisnostima unutar teksta te dovršavanje rečenica. Za procjenu sposobnosti modela koristili su se skupovi podataka LAMBADA [32] i HellaSweg [33]. LAMBADA se koristi za predviđanje posljednje riječi u rečenici na temelju konteksta cijelog odlomka. Kako bi model uspješno riješio zadatke na LAMBADA skupu podatka ne može samo koristiti lokalne informacije, već mora razumjeti širi odlomak. HellaSweg sadrži zadatke u kojima model mora odabrati najvjerojatniji nastavak zadane rečenice između više ponuđenih opcija. Distraktori su posebno osmišljeni kako bi zbunili model pa je za uspješno rješavanje potrebno ne samo razumijevanje jezika, već i logičko zaključivanje na temelju konteksta.

U svim provedenim testovima xLSTM pokazala se bolja od LSTM te je postigla rezultate usporedive s popularnijima transformativnim modelima.

# 4 Primjer RNN i LSTM za generiranje teksta

## *4.1 Šifriranje teksta*

Prije testiranja mogu li RNN i LSTM generirati smisleni tekst zadajemo im jednostavniji zadatak učenja šifriranja teksta. Kao primjer uzet ćemo Cezarovu šifru i Cezarovu šifru s varijabilnim pomakom.

Prema povijesnim izvorima, Cezarova šifra datira iz vremena Rimskog Carstva, kada je Julije Cezar koristio ovu tehniku za sigurnu komunikaciju u vojne svrhe. Šifriranje funkcionira tako da se svako slovo u poruci zamijenilo drugim slovom koje se nalazi proizvoljan broj mjesta dalje u abecedi. U kodu dostupnom na [25] koristi se pomak od 3. Tako će mreže morati naučiti da je slovo 'a' potrebno zamijeniti slovom 'd', slovo 'b' slovom 'e' i tako dalje. Tekst na kojem su mreže trenirane nasumično je generiran za svaku epohu i sastoji se od 100 uzastopnih slova.

Obje mreže naučile su u samo 10 epoha ispravno šifrirati poruku. Kod je moguće isprobati na [37], a niže su slike primjera dobivenih rezultata.

```
Original: CJ_W_EJuFfkOFeNxZBfvIMBPrp.eRrh,K,NcGDDDLBOVhcVVep.CvsYwvAnnOetibgxDIlNL.F-eRCeICGXCKljsL.arXyybIoab
True Cipher: FM_Z_HMxIinRIhQaCEiyLPESus.hUuk,N,QfJGGGOERYkfYYhs.FyvBzyDqqRhwlejaGLoQO.I-hUFhLFJAFNomvO.duAbbeLrde
Predicted Cipher: FM_Z_HMxIinRIhQaCEiyLPESus.hUuk,N,QfJGGGOERYkfYYhs.FyvBzyDqqRhwlejaGLoQO.I-hUFhLFJAFNomvO.duAbbeLrde
```

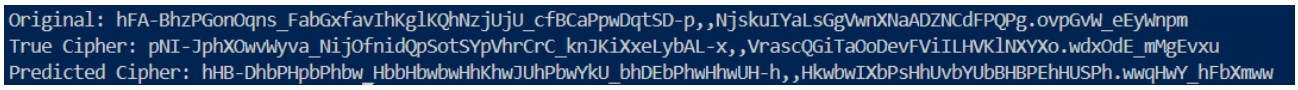
Slika 4.1: Rezultati dobiveni nakon 10 epoha treninga za RNN, Cezarova šifra

```
Original: AfHUXcOBXykJwxNChpdrKYVAYzcZIRzQXGdazxDjuTGLobomTRNroUdioIabwwTZXrebpDHnzJUrrswGyEKOYrQpNVEnQnYvujPl
True Cipher: DiKXAfREAbnMzaQFksguNBYDBcfCLUcTAJgdcaGmxWJOrerpWUQurXglrLdezzWCAuhesGKqcMXuuvzJbHNRBuTsQYHqTqByxmSo
Predicted Cipher: DiKXAfREAbnMzaQFksguNBYDBcfCLUcTAJgdcaGmxWJOrerpWUQurXglrLdezzWCAuhesGKqcMXuuvzJbHNRBuTsQYHqTqByxmSo
```

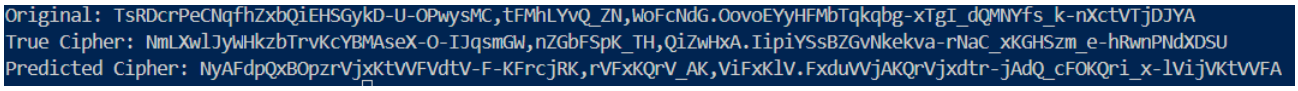
Slika 4.2: Rezultati dobiveni nakon 10 epoha treninga za LSTM, Cezarova šifra

Kako bismo malo bolje testirali razlike u opažanju dugoročnih ovisnosti pogledajmo mogu li obje mreže naučiti Cezarovu šifru s varijabilnim pomakom. Kod ove vrste šifriranja pomak ovisi o prvom slovu poruke. Tako ako je početno slovo nešifrirane poruke 'a' pomak će biti 1, ako je prvo slovo nešifrirane poruke 'b' pomak će biti dva i tako dalje.

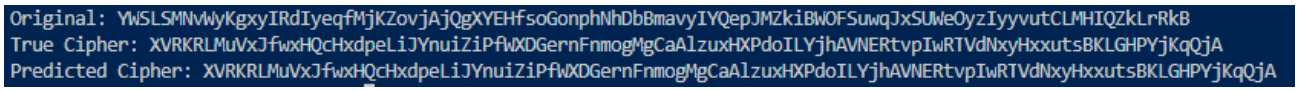
U ovom slučaju LSTM pokazala se brzom u uočavanju dugoročnih ovisnosti te je naučila šifrirati tekst u 20 epoha, RNN i nakon 150 epoha nije davala 100% točne rezultate.




Slika 4.3: Rezultati dobiveni nakon 10 epoha treninga za RNN, Cezarova šifra s varijabilnim pomakom




Slika 4.4: Rezultati dobiveni nakon 150 epoha treninga za RNN, Cezarova šifra s varijabilnim pomakom




Slika 4.5: Rezultati dobiveni nakon 20 epoha treninga za LSTM, Cezarova šifra s varijabilnim pomakom

U slučaju pokretanja koda [37] unutar Google Colab okruženja potrebno je više od 20 epoha kako bi mreže davale točne rezultate. Razlozi za ovo mogu biti različiti, poput razlika u hardwareu na osobnom računalu i Google Colabu (različiti procesori mogu imati različite optimizacije za linearnu algebru) te razlika u načinu nasumičnog inicijaliziranja težina.

## *4.2 Primjer koda za generiranje teksta*

U ovom poglavlju pogledat ćemo primjer koda u programskom jeziku Python za rekurentnu neuronsku mrežu te uz minimalne promjene Long short term memory mrežu. Kod je preuzet s [16]. Svakoj mreži bit će zadani tekstovi različitih duljina na kojima će trenirati tijekom 500 epoha. Na kraju svake epohe mreže će dobiti komad teksta (seed) te će se sama generirati nastavak teksta na temelju naučenog.

```
from __future__ import absolute_import, division, print_function,
unicode_literals

import numpy as np
import tensorflow as tf
```

```
import io
import sys

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.layers import LSTM
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import LambdaCallback
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
import random
import time
```

Najvažnija biblioteka koja je posebno razvijene za strojno učenje i razvoj umjetne inteligencije je Tensorflow. Nije dio standardnog Python paketa te ju je potrebno instalirati. Verzija TensorFlow-a koja se koristi u ovom radu je 2.8.0. Još jedna važna stavka, nekad samostalna biblioteka, a sada dio Tensorflowa, Keras, otvoreno je aplikacijsko sučelje dizajnirano da bude modularno i korisniku lako za čitanje i pisanje. Omogućuje brzu izgradnju i treniranje modela dubokog učenja. S pomoću Kerasa može se definirati arhitektura neuronskih mreža koristeći sekvencijalni ili funkcionalni API. Sekvencijalni modeli prikladni su za linearne zadatke, dok funkcionalni modeli pružaju više fleksibilnosti za složene arhitekture. Jedna od najvećih prednosti korištenja Tensorflow je njegova mogućnost da brine za iskorištenost resursa računala. Biblioteka pokušava optimalno podijeliti računske zadatke koji se obavljaju tijekom treninga na jezgre procesora i grafičke komponente i tako ubrzati proces treniranja neuronskih mreža [38]. Za potrebe ovog rada trening mreža proveden je u docker kontejnerima (eng. *Docker container*) kako bi se osigurala što veća izoliranost treniranja mreža od ostatka sustava [39]. Kontejneru je pri pokretanju dan pristup dvjema jezgrama procesora kako bi se pokušali osigurati što sličniji resursi u svim primjerima radi usporedbe vremenskog trajanja treninga.

```
# Postavljanje standardnog output encodinga na utf-8
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
with open('texts/text.txt', 'r', encoding='utf-8') as file:
```

```
    text = file.read()
vocabulary = sorted(list(set(text)))

char_to_indices = {c: i for i, c in enumerate(vocabulary)}
indices_to_char = {i: c for i, c in enumerate(vocabulary)}

max_length = 100
steps = 5
sentences = []
next_chars = []
for i in range(0, len(text) - max_length, steps):
    sentences.append(text[i: i + max_length])
    next_chars.append(text[i + max_length])

X = np.zeros((len(sentences), max_length, len(vocabulary)), dtype=bool)
y = np.zeros((len(sentences), len(vocabulary)), dtype=bool)
for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_to_indices[char]] = 1
    y[i, char_to_indices[next_chars[i]]] = 1
```

Tekst na kojem mreža trenira nalazi se u tekstualnoj datoteci naziva text.txt. U varijablu `vocabulary` spremaju se svi znakovi iz teksta. Tim znakovima služit će se neuronska mreža pri generiranju vlastitog teksta. Znakovima će se pridružit numerička vrijednost kako bi mreža radila predviđanja s numeričkim podacima, ne tekstualnim. U radu s velikim tekstovima može doći do nestanka radne memorije na računalu te se cijeli program prekida. S obzirom na to da su tekstovi na kojim će se trenirati mreža dugi radit ćemo s manjim dijelovima teksta, odnosno radit će se grupiranje (eng. *batching*). Uz pomoć varijable `max_length` određuje se koja je to duljina teksta koja će se obrađivati po iteraciji. `X` je trodimenzionalno polje koje predstavlja ulazne podatke (sekvencu karaktera iz teksta za trening). Dimenzija `len(sentences)` je broj uzoraka za treniranje, `max_length` odgovara duljini uzorka, `len(vocabulary)` predstavlja broj jedinstvenih karaktera u tekstu. `y` je dvodimenzionalno polje gdje je prva dimenzija `len(sentences)`, a druga `len(vocabulary)`. `y` sadrži sljedeći

znak koji slijedi u sekvenci koju sadrži X.

```
model = Sequential()
model.add(SimpleRNN(128, inputShape =(max_length, len(vocabulary))))
model.add(Dense(len(vocabulary)))
model.add(Activation('softmax'))
optimizer = RMSprop(lr = 0.01)
model.compile(loss ='categoricalCrossentropy', optimizer = optimizer)
model.summary()
```

Uz pomoć biblioteke keras stvaramo sekvencijalni modele SimpleRNN ili LSTM koji će raditi sa sekvencijalnim podacima. Sekvencijalni podaci podrazumijevaju bilo koji tip podataka gdje je poredak bitan, sekvencijalni model je svaki model koji uči na sekvencijalnim podacima i za svaki vremenski korak ima samo jedan ulazni i izlazni vektor. Naredba

```
model.add(SimpleRNN(128, inputShape =(max_length, len(vocabulary))))
```

kreira jedan SimpleRNN sloj s jednim skrivenim stanjem koje sadrži 128 neurona. Parametar `input_shape` definira oblik ulaznih podataka na kojim će mreža trenirati: `max_length` označava broj vremenskih koraka $t_0$, dok `len(vocabulary)` predstavlja veličinu $n$ ulaznog vektora koja odgovara broju različitih znakova koji se javljaju u tekstu. Za aktivacijsku funkciju odabrana je softmax funkcija (2.22). S obzirom na to da program ima zadatak raditi s višeklasnom klasifikacijom (generiranje teksta znak po znak) za funkciju gubitka izabrana je poprečna entropija, koja se još naziva i softmax gubitak. Poprečna entropija mjeri razliku između dvije distribucije - prave distribucije (koja se dobiva iz teksta na kojem mreža trenira) i distribucije koju je predvidio model mreže (distribucija se uzima iz izlaza zadnjeg vremenskog koraka neuronske mreže). Funkcija gubitka računat će se kao:

$$L = H(y, \hat{y}) = -\sum y_i log(\hat{y}_i) \tag{4.1}$$

gdje y označava pravu distribuciju, a $\hat{y}$ predviđenu distribuciju. Cilj postupka treniranja je minimizirati ovaj gubitak, što implicira da bi se predviđanja modela trebala približiti stvarnim ciljnim vrijednostima kako napreduje obuka.

Nadalje, koristi se `RMSProp` [40] (*eng. Root mean square propagation*) optimizer sa stopom učenja od 0.01. Ovdje se koristi pristup postavljanja stope učenja na malu konstantu. Mala stopa učenja obično se koristi kako bi se spriječilo mrežu da prebrzo donosi zaključke, odnosno kako bi se parametri mreže sporije ažurirali. [41]. Općenito, algoritam za optimizaciju uzima gradijent gubitka te na temelju njega odlučuje kako će se ažurirati težine prema pravilu koje je definirano u jednadžbi (2.15).

Nadalje, stvaramo funkciju `sampleIndex` koja će generirati tekst.

```
def sampleIndex(preds, temperature = 1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    expPreds = np.exp(preds)
    preds = expPreds / np.sum(expPreds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

Varijabla `temperature` određuje koliko će nasumičan biti taj generirani tekst. Temperatura je jedan od hiperparametara mreže, odnosno eksplicitno se zadaje na početku i vrijednost joj se više ne mijenja. Mreža će pri generiranju teksta niske temperature od 0.5 češće odabirati riječi s većom vjerojatnošću pojavljivanja u tekstu, što prema literaturi [36] daje predvidljivije i konzistentnije rezultate. Dok za mrežu koja generira tekst s temperaturom 1.0 generirani rezultati će biti balansirani između onih koji se češće javljaju u tekstu za trening i onih koji se rjeđe javljaju. U funkciji `sampleIndex` polje `preds` sadrži vjerojatnosti za generiranje svakog znaka iz vokabulara kao slijedeći znak. Na primjer, ako su u vokabularu znakovi [’a’, ’b’, ’c’] `preds` sadrži koje su vjerojatnosti da taj znak bude idući generiran, na primjer [0.1, 0.2, 0.4].

Nadalje, kroz kod se mogu pratiti matematičke operacije poput logaritmiranja, dijeljenja s temperaturom i potenciranja koje se primjenjuju kako bi se došlo do konačnog rezultata. S pomoću funkcije `np.argmax(np.random.multinomial(1, preds, 1))` izabire se znak s najvećom vjerojatnošću nakon provedenih računskih operacija.

Promotrimo sada funkciju `on_epoch_end`:

```
def on_epoch_end(epoch, logs):
```

```
    sys.stdout.write(f'----- Generating text after Epoch: {epoch}\n')
    start_index = random.randint(0, len(text) - max_length - 1)
    for diversity in [0.5, 1.0]:
        print(f'----- diversity: {diversity}\n')
        generated = ''
        sentence = text[start_index: start_index + max_length]
        generated += sentence
        sys.stdout.write(f'----- Generating with seed: "{sentence}"\n')
        sys.stdout.write(generated)
        sys.stdout.flush()
        for i in range(400):
            x_pred = np.zeros((1, max_length, len(vocabulary)))
            for t, char in enumerate(sentence):
                x_pred[0, t, char_to_indices[char]] = 1.
            preds = model.predict(x_pred, verbose=0)[0]
            next_index = sample_index(preds, diversity)
            next_char = indices_to_char[next_index]
            generated += next_char
            sentence = sentence[1:] + next_char
            sys.stdout.write(next_char)
            sys.stdout.flush()
        print("\n")
```

U funkciji `on_epoch_end` pokreće se treniranje mreže. Za svaku epohu mreža uzima tekst određene maksimalne dužine te samostalno generira predviđeni nastavak teksta. Rezultati se generiraju za temperature 0.5 1.0.

Naposljetku:

```
#Ispis napretka nakon svake epohe
print_callback = LambdaCallback(on_epoch_end=on_epoch_end)

#Spremanje modela samo kada se model poboljša (gubitak se smanji)
filepath = "model.keras"
checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1,
    save_best_only=True, mode='min')
```

```
#Smanjivanje stope učenja ako se gubitak ne mijenja
reduce_alpha = ReduceLROnPlateau(monitor='loss', factor=0.2,
    patience=1, min_lr=0.001)
callbacks = [print_callback, checkpoint, reduce_alpha]

'''Pozivanje treninga modela na podacima X i y, 500 epoha, koristeći
grupe podataka veličine 128'''
model.fit(X, y, batch_size=128, epochs=500, callbacks=callbacks)

#računanje koliko je vremena bilo potrebno za trening
end = time.time()
elapsed_time_hours = (end - start) / 3600
print("The time of execution of the above program is:",
    elapsed_time_hours, "hours")
model.summary()
```

Prije početka treniranja i na samom kraju poziva se funkcija *model.summary()* koja prikazuje arhitekturu modela kao na slici 4.6.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 128)               23552

 dense (Dense)               (None, 55)                7095

 activation (Activation)     (None, 55)                0

=================================================================
Total params: 30,647
Trainable params: 30,647
Non-trainable params: 0
_________________________________________________________________
```

Slika 4.6: Primjer izlaza za metodu model.summary()

Na slici je prvo je naveden tip modela, u primjerima iz ovog rada svi modeli su sekvencijalni. Zatim slijede podaci o tipu slojeva modela, izlaznom obliku i broj pa-

rametara. Izlazni oblici u svim primjerima u ovom radu oblika su `(None, num)` te predstavljaju dimenzije pojedinog sloja. `None` predstavlja veličinu serije, postavlja se kao objekt `NoneType` klase automatski. Ovaj tip podataka označava da je varijabla prazna i osigurava fleksibilnost i mogućnost dinamičkog dodjeljivanja vrijednosti tijekom treniranja [42]. `num` predstavlja broj neurona unutar skrivenog stanja modela te se zadaje pri kreiranju modela metodom `model.add()`. Broj parametara za SimpleRNN sloj računa se kao:

$$P_{\text{RNN}} = H \times (n + H + 1) \tag{4.2}$$

U LSTM sloju koji koristimo uzima se prema jednadžbama (3.1), (3.4) tako da se broj parametara za LSTM sloj računa kao:

$$P_{\text{LSTM}} = 4 \times p \times (n + p + 1) \tag{4.3}$$

gdje n označava broj ulaznih jedinica, u ovom slučaju to je broj znakova u vokabularu `len(vocabulary)`. *p* označava broj neurona unutar skrivenog stanja mreže.

U slučaju biblioteke keras formule za ono što smo definirali kao RNN (i LSTM) razdvojeno je na dva sloja - potreban nam je još i takozvani gusti (eng. *dense*) sloj. U gustom sloju broj parametara računa se prema jednadžbi (3.8):

$$P_{dense} = q \times (p + 1) \tag{4.4}$$

ili u slučaju RNN prema jednadžbi (2.3)

$$P_{dense} = p \times (H + 1) \tag{4.5}$$

Iz jednadžbi (4.2) i (4.3) vidljivo je da će LSTM uvijek imati veći broj parametara za isti broj neurona. Jedna od stvari na koje će se obratiti pažnja pri treningu je da RNN i LSTM koje treniramo imaju isti broj parametara. Kako bi se to postiglo varirat će se zadani broj neurona RNN mreže.

Model iz primjera sa slike 4.6 ima tri sloja: SimpleRNN sloj s izlaznim oblikom (None,128) i 23552 parametara, gusti sloj s izlaznim oblikom (None, 55) i 7095 parametara te aktivacijski sloj s izlaznim oblikom (None, 55) i 0 parametara. Ukupni

broj parametara je tako 30647 i svi parametri se mogu trenirati, odnosno vrijednosti težina i pristranosti ažuriraju se tijekom epoha.

Jedina razlika između koda za LSTM i RNN mrežu je prilikom uvoza biblioteka te kreiranja mreže.

```
from keras.layers import LSTM
i
model.add(LSTM(128, inputShape =(max_length, len(vocabulary))))
```

## *4.3 Generiranje teksta - dobiveni rezultati*

U sljedećem zadatku mrežama su dani tekstovi na engleskom jeziku. Tekstovi na kojima je testirana mreža je Gavran od Allana Edgara Poe [43], Hamlet od W. Shakespeara [44] i Zločin i kazna autora F. Dostojevskog [45]. Trening je proveden na 500 epoha. Duljina vokabulara u navedenim tekstovima je 55, 72 i 83.

Kako je količina teksta koja se generira velika, kod je pokrenut putem komandne linije kao "`python nazivPrograma.py > nazivTekstualneDatoteke.txt`". Svi rezultati koje ispisuje naredba `sys.stdout.write` ispisat će se u tekstualnu datoteku te je tako lakše spremati podatke za kasniju analizu. U nastavku su prikazani rezultati prve epohe, epohe s najboljim rezultatima te epohe između prve i najbolje kako bi bilo vidljivo kako neuronska mreža napreduje. Cjeloviti rezultati dostupni su na [25]. Za rezultate prikazane u tablici koristio se python kod dostupan na [25], `grammer_check.py` i `word_counter.py`. Funkcija `grammer_check` vraća broj točaka na kraju rečenice te broj velikih slova nakon točke. Za poemu Gavran također provjerava je li na početku svakog novog stiha veliko slovo. Funkcija `word_counter` koristi biblioteku nltk koja u sebi sadrži listu svih engleskih riječi, za generirani tekst provjerava koliko se riječi nalazi u toj listi. Provjera postotka engleskih riječi je napravljena za posljednjih 50 epoha te su rezultati prikazani u tablicama.

### 4.3.1 Gavran - RNN

Epoch 1/500

—— diversity: 0.5

—— Generating with seed: "Tis some visitor entreating entrance at my chamber door Some late visitor entreating entrance at my "

Tis some visitor entreating entrance at my chamber door Some late visitor entreating entrance at my

bldbndblbldlblhlbbldbbrddbbgdllldslrhrlndnldldldlhlblrbdlnlbblblnblrrlbhhbllb lbdlrnb drdbhrllrrnbbbdlblhrlnlhlbbllllbhlb,hglbbdllbnlbrblbblynldbnbnnlblrl rrbllnd bbllbbdhlldblblbb lldbllbd,lrlbblybblabdhlbrbllrbdnllllrlblbngbllblhrbllllll rlrbdnlbng hellll dllbblslbbllkylbllllbllllblhlbbhhnhlwl rhbbtlbdbglndblglbsrlllblbnllrlldlbdwlllrldlrlllblllldbbrlb lbdllllhblbll rlldldhblrbkdlbhdhdrdlldllrdrbbbbrwlldl

—— diversity: 1.0

—— Generating with seed: "Tis some visitor entreating entrance at my chamber door Some late visitor entreating entrance at my "

Tis some visitor entreating entrance at my chamber door Some late visitor entreating entrance at my

bgrdbwgrDhicbrnrcl,rlllkdpwsrnbhlwnhi rhhwhlrdrc lbwhellrlkrwblbrltrnpasdrrbghyrl dkdfrblHkllw ydSdbnnyrbhwbhbnhllebwbnhldbbrrddrgdnd wuglwhbrhIhhlbrdlbbglbrblnhrdblywrgbhndl ddbdkdndbilbldlrbdnrrllbdlrbrbeddwrbdNreNls, hhsilhhdgdllnbdkdbsohkwdhbbb Pdlrbgllknplrbllowlndlb dlIbucnnbbthllldbedhhehlgsrhlerdlllghrhhbllhlbrhrhbrghpbb lraryhlskDlnllslylblnb,lpsd,bbbSlslrrnbrhllshbhdudhihblbrlDkhd klbbhd

**Epoch 1: loss improved from inf to 3.81615, saving model to model.keras**

Epoch 184/500

—— diversity: 0.5

—— Generating with seed: " the Nightly shores

Tell me what thy lordly name is on the Nights Plutonian shore!

Quoth the Raven "

the Nightly shore
Tell me what thy lordly name is on the Nights Plutonian shore!
Quoth the Raven se sour wher seatent oress aplore.
Tisthe more
Therather dher thare my ther my me shere sher hesatin by my an the seem moug thet me thee on d or more.
Qhe morg mord the sist ond thor more
Ting me I more Iore am then berererererame sa se seame my orethee mererever saen my my me or verereden la smy thind by t me ssing, whesseaml ther on er, I fore by thather moud shin the saer fher more my ther them l

—— diversity: 1.0
—— Generating with seed: " the Nightly shores
Tell me what thy lordly name is on the Nights Plutonian shore!
Quoth the Raven "
the Nightly shores
Tell me what thy lordly name is on the Nights Plutonian shore!
Quoth the Raven org, salt a mj my mirt top shyd ore diat me sh sist ohe nathing Dveresmy tateism ;akeraln d teorderhenter mo, boarenot threrwiso Lere anont toing sait lhass on byen my bhat anggreremordplout molsm, thes nore.
Thupant by sing bead ffllltiot my s lhe loat wout ondenease maraissat erder,
Thelson mÇö seerekang owrentll ver ond wkeasHerer stor bart sukd Thing In bheriqhing miame samollle horg neaot thel
**Epoch 184: loss did not improve from 1.64886**

Epoch 369/500
—— diversity: 0.5
—— Generating with seed: "guiling all my fancy into smiling, Straight I wheeled a cushioned seat in front of bird, and bust an"
guiling all my fancy into smiling,
Straight I wheeled a cushioned seat in front of bird, and bust and tare.aver, madch whan, beraist, thist whanct on bore sher de sat, I, whe it, thesished e

vapp, be bist a saist
what the astiot by cher thethe astin bevermong ouot ber, baiston tore! omowthen, whas tare s mo the caven detait byve more.
Thch the alout beraisg,
Shed theaist, Ieving,
Shatt thea thet ongwhare ane,
Tertint, the aist,
She toing,
She toag, f mait beve more.
Tacs e berthy berais bore!d

—— diversity: 1.0
—— Generating with seed: "guiling all my fancy into smiling,
Straight I wheeled a cushioned seat in front of bird, and bust an"
guiling all my fancy into smiling,
Straight I wheeled a cushioned seat in front of bird, and bust and tareat, and, mare.
Decy ned, Iad,
Sh
,are„
Wich, whd n shid theaist f blaker,
Ded,ien,
I d,ansthing, Iait be sasthat aa m ec aastore.
Leapit, whame seea dis„
Shrdet, thatc rescere ciacb hed, iagthe tait, bemoubtg thent about d saing,!Shate sasteing beve aberd fast me vee of nt I seeviscores, foig, Ihake famt lnen thet ongwh, eamy thin ch anch shiand devba curet,ing,
I H,ish f shed,
Oncbered,de
**Epoch 369: loss did not improve from 0.15271**

U rezultatima je moguće uočiti napredak između prve epohe i najbolje epohe.

Dok su u prvoj epohi generirana slova puno ponavljaju, u najboljoj epohi generiraju se i slova i rečenični znakovi poput zareza, točke i uskličnika. Za diversity 0.5 mreže generira tekst sličan poemi, sa zarezima i točkama na kraju, ali mreža nije uspjela generirati smislen tekst.

### 4.3.2 Gavran - LSTM

Epoch 1/500
—— diversity: 0.5
—— Generating with seed: "ed, weak and weary,
Over many a quaint and curious volume of forgotten lore
While I nodded, nearly n"
ed, weak and weary,
Over many a quaint and curious volume of forgotten lore
While I nodded, nearly nsaeoorms aramarrrnheeamotorodcti roei.eoss tt ree nohe tntht erd a rl,nrnnoaroaee ahosrheegpoo arnasree eo aeaoomesdsade s or em eetlibeer eee relhorrerpec
he lldtreaanaed daedns e,asroia dmo ia tlhdnntteadhartat r eow eaodeesm oorenrrel waoraeav sorkaearware nrtasernesrepc bee eeaodhoor tnoioe erepr dieeom,dde ae e,t ee,enstnd l o sdoeterore too st emteedlrou osddseia,rrno gttor.aeaed r ns

—— diversity: 1.0
—— Generating with seed: "ed, weak and weary,
Over many a quaint and curious volume of forgotten lore
While I nodded, nearly n"
ed, weak and weary,
Over many a quaint and curious volume of forgotten lore
While I nodded, nearly nrsrreodo!aeecaatrargetatn n y d,ÇÖne leswrdo ohya-oftthlathygds,dohkmTw! nllp.noat.aaepi heht aa sveinar,aI
thdldisdf!odebL!eiothrd yierInp.rodCmupopHvae osnmapeRlmmttfwsoiom mnstsomkhucufo,tneppoondasibtl

vo,usgea midtbI.sOshaltocomt phl nd yn,otmrmsmlrnrweooraormSme,s rnqdnm.rLdh
temharmyk!,rw rro uyaduogorassnhg rjkeGeruaOrhGisoaoitnrh
ntfuHmfseeurrcmyosrdl yir
t,a mwGrcr,ctmg , rertbhoog
**Epoch 1: loss improved from inf to 3.40158, saving model to model.keras**

Epoch 293/500
—— diversity: 0.5
—— Generating with seed: " back into the tempest and the Nights Plutonian shore!
Leave no black plume as a token of that lie t"
back into the tempest and the Nights Plutonian shore!
Leave no black plume as a token of that lie tor d vrdere
Ter,len the Raven seeken wher talp be oare
Teuttey wisken myre fllen bore! of et ly berd, th sher moree
loat or or sore,
Teast ea s ond then syen whit end thll the hase ass sthet of bard on what ng bore
Os out or foller more.
Tech ly thit the aissessseeng,oply then sher lave sore.
Tecs lt e visc ardd the Raven sy Heuile, that the orelsser,
Lad the vioken of sore tore.
Prdore.
Prd

—— diversity: 1.0
—— Generating with seed: " back into the tempest and the Nights Plutonian shore!
Leave no black plume as a token of that lie t"
back into the tempest and the Nights Plutonian shore!
Leave no black plume as a token of that lie tor d vist a dhetrenly we stoll, ind

then my that ne woaton ddd on ghtlyec morderr
Thessere, aisdey cahe asd end teun the aises oy what thee assly gad brra.
Arc ent I shere more.
Qeast yais eng thet my Hham hev se I the Raven whst wh sher,
Techlly bere ond the asderseilet byake bore, on ohe aise wn chame t footen of lais the aised sa ghur my ware tore oa coremase or loale assee mi g,
Titt en ing
**Epoch 293: loss improved from 0.00049 to 0.00048, saving model to model.keras**

Epoch 500/500
—— diversity: 0.5
—— Generating with seed: "its ghost upon the floor.
Eagerly I wished the morrow vainly I had sought to borrow
From my books su"
its ghost upon the floor.
Eagerly I wished the morrow vainly I had sought to borrow
From my books sure.
Tecsten the oad tever of eo e! lere.
Prchettey I shed me that nh berais what shev more
TTeatt the Raven boven whepper d the asee, ly that shek the
seessey, and the and barting borr!
Tea store Iist wnd beso ong ttren syel what the osterr il ther I sctreee sshere out ther ssere!more.
PAlouttoe vist on wha smore,
Teustly I oked se ghat en boven of eame dapen warping
Dithem my the seerstere

—— diversity: 1.0
—— Generating with seed: "its ghost upon the floor.
Eagerly I wished the morrow vainly I had sought to borrow
From my books su"

its ghost upon the floor.
Eagerly I wished the morrow vainly I had sought to borrow
From my books sure.
Aecsstt e veor,
y whrtle visp tast ord asd my Hadea nd w, then bore!
Oagt thy isoe soree sserplaed or whla the vast on whrellene dast or orresseere
Ohes sa hevemy thit sy chame wesore, and this eahe se or,
Shes sy kevore!
loacp tree ssery,
I chhet eng, wirrl
en bhat me ghke shere.
Prcore.
Prklyttey asserrllate fout that nd woat ne doven theken Dhtor more.
Prfsetty a odd ne upll the hssen
**Epoch 500: loss improved from 0.00024 to 0.00024, saving model to model.keras**

U najboljoj epohi, u usporedbi s početnim epohama, generirani tekst više nalikuje na poemu na kojoj je mreža trenirana; na početku stiha su velika slova, javljaju se česte riječi iz teksta kao *'Raven'* i *'Nevermore'* što se nije moglo uočiti kod rezultata rekurentne mreže. Konačni tekst nije smislen.
Analiza generiranog teksta u svim primjerima ostavlja loš dojam; i u posljednjoj epohi generirane engleske riječi su kratke, poput *'more', 'or', 'my', 'then'*, engleske riječi s više od pet slova se te su riječi međusobno nepovezane. U svakom generiranom tekstu može se uočiti barem jedna riječ u kojoj se slova više puta ponavljaju, na primjer *aissessseeng*. Jedina višesložna riječ koja se pojavljuje, a da nije *raven* ili *nevermore* je *warping* u 121., 178. i 500. epohi.

U svim primjera možemo uočiti mali gubitak, čak najmanji u usporedbi sa svim kasnijim rezultatima. Kao što je vidljivo na grafu 4.7 gubitak teži u nulu te je gotovo konstantan. Zanimljivo je primijetiti kako LSTM ima nagli pad gubitka u prvih 50 epoha koji je do kraja treninga konstantan, dok kod RNN gubitak kontinuirano pada tijekom 350 epoha, odnosno 150 epoha za 268 neurona te tada postaje konstantan. Tu možemo vidjeti da se trening LSTM mreže mogao zaustaviti već nakon 50 epoha.

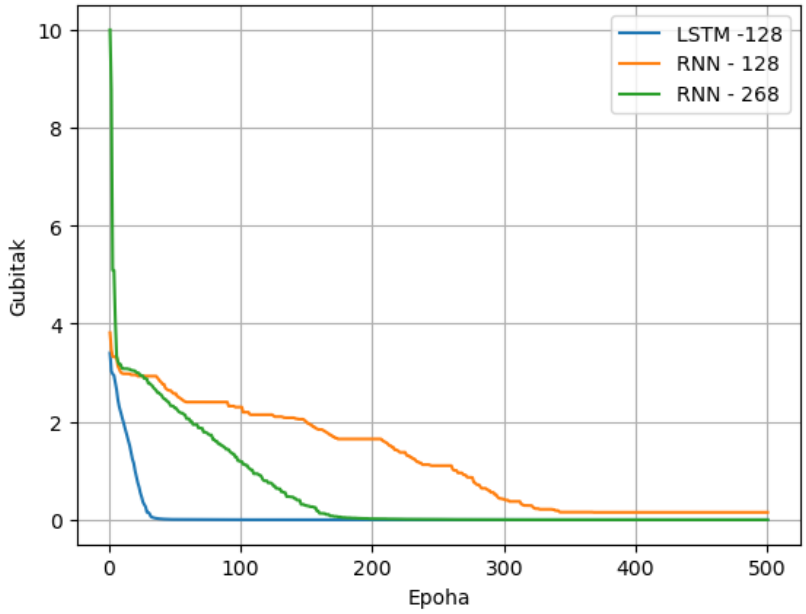
Za jedan od razloga za loše rezultate također bi se mogao navesti prevelik broj neurona. Prevelik broj neurona može dovesti do prenaučenosti ili neuočavanja važnih obrazaca. Iz tog razloga isti trening proveden je i s mrežom sa samo tri neurona te je postotak generiranih engleskih riječi nešto veći. Rezultati ostvareni s tri neurona dostupni su na [25]. Za uspješan trening mreže ključan je opširan skup podataka te može čak i biti važniji od samih postavki parametara mreže. Trening je ponovljen i za RNN sa 7 i 268 neurona kako bismo usporedili rezultate kada RNN ima približno sličan broj parametara kao i LSTM.

| broj riječi: 1087 | RNN | LSTM | RNN |
|---|---|---|---|
| broj neurona | 3 | 3 | 7 |
| trajanje /h | 9.20635 | 9.80828 | 8.43884 |
| broj parametara | 397 | 928 | 881 |
| postotak engleskih riječi | 55 $\pm 4$ | 60 $\pm 4$ | 53 $\pm 5$ |
| % velikih slova na početku | 72.1 | 64.4 | 71.6 |

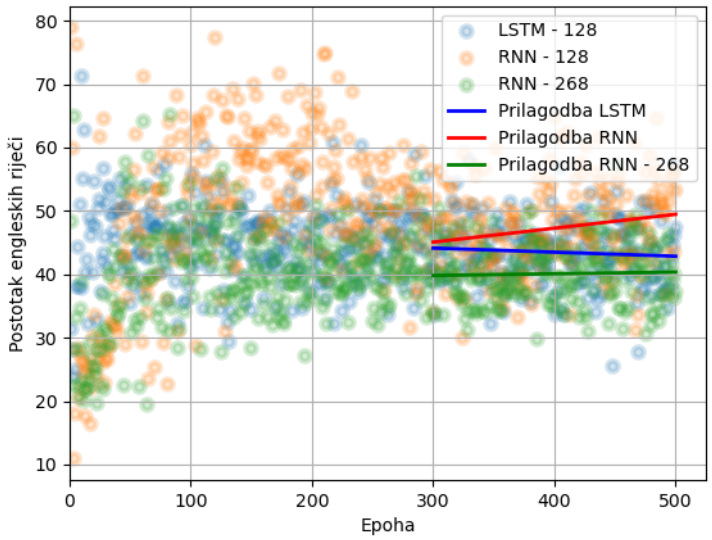
Tablica 4.1: Usporedba rezultata za poemu Gavran

| broj riječi: 1087 | RNN | LSTM | RNN |
|---|---|---|---|
| broj neurona | 128 | 128 | 268 |
| trajanje /h | 10.35407 | 10.31885 | 8.59091 |
| broj parametara | 30,647 | 101,303 | 101,627 |
| postotak engleskih riječi | 49 $\pm 7$ | 43 $\pm 6$ | 40 $\pm 5$ |
| % velikih slova na početku | 83.3 | 74 | 69.8 |

Tablica 4.2: Usporedba rezultata za poemu Gavran




Slika 4.7: Usporedba funkcija gubitka za LSTM i RNN na primjeru teksta Gavran




Slika 4.8: Usporedba postotka engleskih riječi za LSTM i RNN na primjeru teksta Gavran

U tablici se može uočiti da je LSTM mreža s tri neurona ima najveći postotak stvarnih engleskih riječi. Na grafu 4.8 vidljivo je da RNN sa 128 neurona ima najbolji prosječan udio stvarnih engleskih riječi dok RNN s 268 neurona ima manje postotke u usporedbi s ostalim mrežama. RNN sa 128 neurona također ima najveće oscilacije u postotku engleskih riječi od epohe do epohe. Kod nje je za naizgled izuzetno uspješne rezultate na početku utjecala činjenica da biblioteka "nlkt" broji osobnu zamjenicu ’I’ i neodređeni član ’a’ kao engleske riječi.

### 4.3.3 Hamlet - RNN

Epoch 1/500

—— diversity: 0.5

—— Generating with seed: "metals base,

Shows itself pure; he weeps for what is done.

KING CLAUDIUS

O Gertrude, come away!

The "

metals base,

Shows itself pure; he weeps for what is done.

KING CLAUDIUS

O Gertrude, come away!

The tlteet

yti aheeaseeee mStee wit me s ese nseatatonealewiean wedoo aahrhanettee ee seeeayotot aae toelt se aotoae, T t to aeoms aaaeeasethameeeteleeeemeet aamestoseneetee toseee teseel tStaneerhee anhllenl neeen lraoalaeaeonnela-nee een tsee tees tee ne eoonteleee s se atete tee le loeetet te teoet te teee m t t aee meyrsteiteeloieoamt soteeeote a wh,

ns te aeeotoboeee ar lo whee teseee ae ase

—— diversity: 1.0

—— Generating with seed: "metals base,

Shows itself pure; he weeps for what is done.

KING CLAUDIUS

O Gertrude, come away!

The "

metals base,

Shows itself pure; he weeps for what is done.

KING CLAUDIUS

O Gertrude, come away!

The

Rge;so

,te ansgoeelteiwHacerr tphar-,ewoey ocsneosmtost or oo aiaccnh
.ah ePse
enmjr
nethsa,acHte, as gsenrle oiiad fe
e pfaoR, wouanase
kkns rt
.lghsahfPwanthtauro elle
eehase
GhsfonLrl
teo
wetRm lownins tms r eehny mhlnl-naOFa;henhnnibweeRM;owe!mcmes hoet-nth tse w:esao
i
DQalpt ftobw'ee,
6s ne seoeytsneamatoReenbasi gyhkaPtor
ElsSPuiiEseoia wHh
t elgeitebbetoe l
.anoeer anayhetare sHGfeg hme
**Epoch 1: loss improved from inf to 3.72120, saving model to model.keras**

Epoch 235/500
—— diversity: 0.5
—— Generating with seed: "eans to the king:
they have letters for him. Ere we were two days old
at sea, a pirate of very warli"
eans to the king:
they have letters for him. Ere we were two days old
at sea, a pirate of very warling your wace seet to comereme to the say, mote here? Exha the Preat?
ont it a tre out of the cones the corent of lord a will defther owr mowr will yound of han our and it shale be porse, and the forate, the stores a father makes sing o'e tour dinge and the dese.

HAMLET

Ole to the murd offered stound: the haud dood deft then and my dot spemplote sere. There would and mant to still and lest to de po

—— diversity: 1.0

—— Generating with seed: "eans to the king:

they have letters for him. Ere we were two days old

at sea, a pirate of very warli"

eans to the king:

they have letters for him. Ere we were two days old

at sea, a pirate of very warling, I at e dain a, frovent

The done a muad,

To nit moser

The so'd rellle the bedser,,

ey the kings, at po,

Ly to goist enensis in at e sayraged, stithn herse.

On these chaned:

Thelingard make ouchin] yowrte

To murd the, thee to serpown there

Ast patere.

Exit

Where that his myed, sir.

HAMLET

Is gony camnont lomd and sorker with whith it.

Hor this itce is your of.

WheeG hind in it reothers, vinly 5y

**Epoch 235: loss did not improve from 1.52040**

Epoch 469/500

—— diversity: 0.5

—— Generating with seed: "LONIUS

That's good; 'mobled queen' is good.

First Player

’Run barefoot up and down, threatening the "

LONIUS

That’s good; ’mobled queen’ is good.

First Player

’Run barefoot up and down, threatening the gracksters.

Exit

GUILDo!

Anm GUILDENS

Becour astoug and and so mock,

I kimered that it o re somonks for that the ourtlan,

A the wind head not him himse cengert prose hirh, my lore„ a mone ar sallous this is our falle

Dike we may his singers and miss agains of the kiss of majents we agationd.

Othe murds.

Fors cughtere seller it to chenesur:

I so are meriter haven a chice form and sincy, and with

—— diversity: 1.0

—— Generating with seed: "LONIUS

That’s good; ’mobled queen’ is good.

First Player

’Run barefoot up and down, threatening the "

LONIUS

That’s good; ’mobled queen’ is good.

First Player

’Run barefoot up and down, threatening the voin,

Will yourselutod ime coulian

warle

I: ’therh.

EThon at o’ Hamunc; your love

Them the racer he shack you sief,

But; enet sirm frough the rasged, syen Clesped,
That hardser wuly may bett to him emeedass of Oury be disuse;
Fete evend throwite art?
MARvRUDERAMA, I sof yoo,
Adr Tuamimear allse; seang
-ay by withurt mad, hot as as are, my his and sincun's for'ton!
Had I gouch sturks and his meten
**Epoch 469: loss improved from 1.44820 to 1.44784, saving model to model.keras**

Rezultati rekurentne mreže na većem tekstu doimaju su bolji, iako do 500. epohe tekst još nije smislen. Značajno je za uočiti da je mreža donekle naučila do zadnje epohe imena likova te da ih je potrebno napisati velikim tiskanim slovima. Mjestimično nakon imena lika nastavi pisati tekst u novom redu, a mjestimično odmah nastavi generiranje teksta iza imena lika, bez prijelaza u novi red. Može se uočiti i da postoje nepotrebni prekidi tekst lika kako bi se ponovno napisalo ime istog tog lika.

### 4.3.4 Hamlet - LSTM

Epoch 1/500
—— diversity: 0.5
—— Generating with seed: "e, against the Polack:
With an entreaty, herein further shown,
Giving a paper
That it might please "
e, against the Polack:
With an entreaty, herein further shown,
Giving a paper
That it might please has no wing no meter and be his the fars and and and we the mave, to has to hat the thas and the rat is in thes the thas s and not hat prall and my l fat the and that has that and I an sham you han the thas the

wave and on has my the mind has as and it and a an and in ros so hav sert, and sale frat hat un and je par the fams and this him the far hat him wive thes my lor, and hat hou se has it as i

—— diversity: 1.0
—— Generating with seed: "e, against the Polack:
With an entreaty, herein further shown,
Giving a paper
That it might please "
e, against the Polack:
With an entreaty, herein further shown,
Giving a paper
That it might please and in qmee gig to this snilane fre and and hit
Brtery for the,
Pamarstow the what I- rars LERy o pratite- that brit resceer'l carlsi; ma ruofbe: muct notes Hacf il mudy tncr loa st. Doll be crten: maingervy , he rn hot thim mull wis f. Bot Leat, no tinler snernkT ow cepeave an t toe. Goon:
Endive inE
Bu by be hag tokefur mereng you huq, of t is abde the me, jo thif farinve ard the deas ler vak th
**Epoch 1: loss improved from inf to 2.64278, saving model to model.keras**

Epoch 239/500
—— diversity: 0.5
—— Generating with seed: "eenly,
In hugger-mugger to inter him: poor Ophelia
Divided from herself and her fair judgment,
Witho"
eenly,
In hugger-mugger to inter him: poor Ophelia
Divided from herself and her fair judgment,
Without and pertate the what hat he

gron,
That wo do put free are not art the brave ne.
Exit
And it now rempord.
HAMLET
Upown hath shall thuped of clow;
And with sume booty with his majesty.
Enter LORD POLONIUS
You thlleave the dowo and I conds the restion
That shily and my sord the wellon’d merer: and wreak,
Playersh six hirce, or set I setire him are do
pracion how ablent
till my prase have heardall

—— diversity: 1.0
—— Generating with seed: "eenly,
In hugger-mugger to inter him: poor Ophelia
Divided from herself and her fair judgment,
Witho"
eenly,
In hugger-mugger to inter him: poor Ophelia
Divided from herself and her fair judgment,
Without your Vuch and flime off incent,
To comm’nd chanied,
That think in great wardoun for the brosh of a most dellie.
First Clown
IA did fore break no, go with his infat,
The ochould no of his sing,
So be brothers of Denmark be to meavior:
I pray you, . Singer what Laers?
HAMLET
Not anbassally at winn’s?

GUILDENSTERN

How most wellonA. BettHand efersed that did reed.

MARCELLUS

So thou hear a fadh!

ROS

**Epoch 239: loss did not improve from 0.01030**

Epoch 478/500

—— diversity: 0.5

—— Generating with seed: "seen this hot love on the wing–

As I perceived it, I must tell you that,

Before my daughter told me"

seen this hot love on the wing–

As I perceived it, I must tell you that,

Before my daughter told me did;

And, and dread, and burn on the hirse of no me, in grantly.

The stoof of the fiever: I am a more, my lord,

Or to more the voise him and exarth, show kill'd canjue

To creat of the amber in phrice, a whill: The sun; of the welfary look,

what is to my parrantly.

MARCELLUS

Lad, il't

In my heaves be come, my lord, with his loves. OPHELIA

You are more, HAMLET

Hor see it it be not are brostly that

—— diversity: 1.0

—— Generating with seed: "seen this hot love on the wing–

As I perceived it, I must tell you that,

Before my daughter told me"

seen this hot love on the wing–

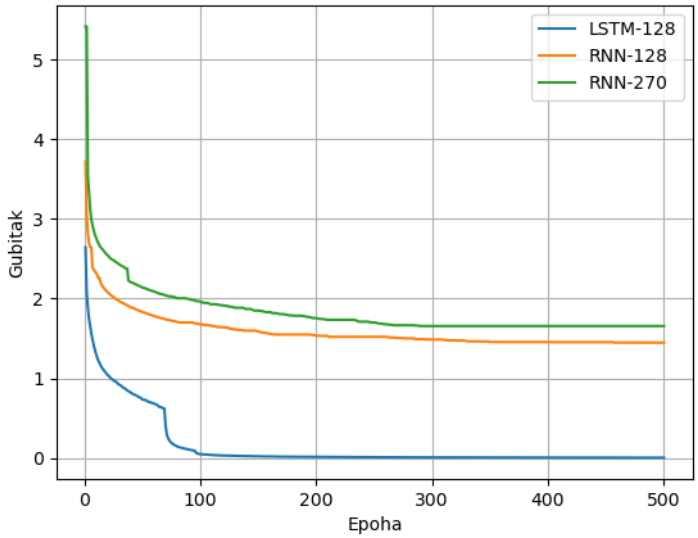
As I perceived it, I must tell you that,

Before my daughter told me. Sect, at shat wouldst with sure
And part and bertrpesty against your have me.
LORD POLONIUS
I am bother hand a chooding well, and the
powor in the fretcry shat where no more by the frother patce.
Hor SOENE Amono we awe meakndos
of your could new, whill has seeme:
The insport the dueb and his courige,
To we azains, and or that your lask come; good friends;
For furst a king;
And shortigh starget m
**Epoch 478: loss improved from 0.00388 to 0.00368, saving model to model.keras**

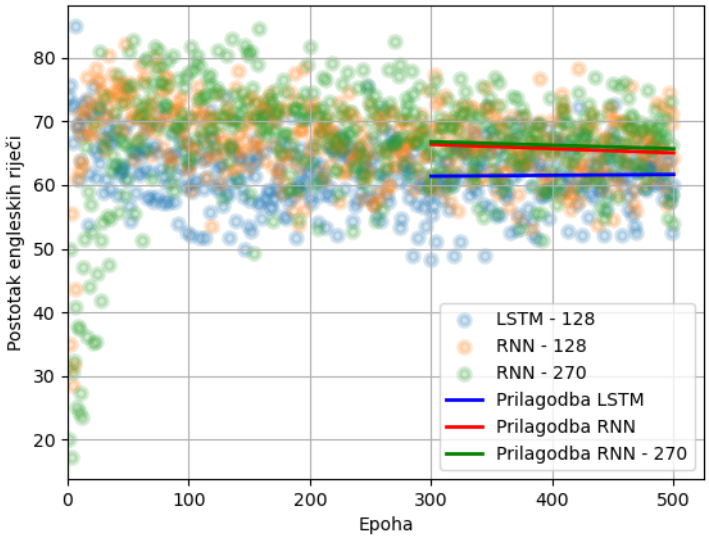
I kod LSTM mreže može se uočiti napredak. Tekst nije smislen, ali postoji struktura slična kao u originalnom tekstu za treniranje. Mreža je naučila imena likova i kako treba nastaviti tekst u novom redu nakon velikim tiskanim slovima napisanog imena lika te da nakon riječi "exit" također treba napisati ime lika velikim slovima. Udio stvarnih riječi iz engleskog jezika čini više od polovice generiranog teksta. Gubitak kod svih mreža kod otprilike 150 epohe postaje konstantan. Rezultati za rekurentnu mrežu s više neurona ne pokazuju značaj napredak u odnosu na istu mrežu s manje neurona što se tiče smislenih engleskih riječi, rezultati su jednaki kao i kod mreže RNN sa 128 neurona.

| broj riječi: 32338 | RNN | LSTM | RNN |
|---|---|---|---|
| broj jedinica | 128 | 128 | 270 |
| trajanje /h | 6.07205 | 8.91245 | 11.964655 |
| broj parametara | 35,016 | 112,200 | 112,122 |
| postotak engleskih riječi | 66 $\pm 5$ | 67 $\pm 4$ | 66 $\pm 5$ |
| ispravno ime like (velika slova, novi red) | 86% | 94% | 97% |

Tablica 4.3: Usporedba rezultata za Hamleta




Slika 4.9: Usporedba funkcija gubitka za LSTM i RNN na primjeru teksta Hamlet




Slika 4.10: Usporedba postotka engleskih riječi za LSTM i RNN na primjeru teksta Hamlet

### 4.3.5 Zločin i kazna - RNN

Epoch 1/500

—— diversity: 0.5

—— Generating with seed: " time, high time to go. Don t follow me, I beseech you, I have somewhere else to go.... But you go a"

time, high time to go. Don t follow me, I beseech you, I have somewhere else

to go.... But you go at ond in to lou the thott ookith Butth. the tor t tim of thulet cook. Yo you in the chout iim. ou the m ut millino haskelmitt. Yo of of ro in the stily tud thou t oncitth. Bu ind the wam to to he the tit the cou mor.. I Berll the mot ool the the in ho the the the at himett thoutt. the th. And mo merl it me thokerl ot thot of cot mou med cottre cotrmit.. no the cot ot it the ol tom it the ot tou

—— diversity: 1.0
—— Generating with seed: " time, high time to go. Don t follow me, I beseech you, I have somewhere else to go.... But you go a"
time, high time to go. Don t follow me, I beseech you, I have somewhere else to go.... But you go atrul inot it so look. Tachite rou ked, ine fimmittor magler. t rith O ouk and dem outh tou lot. ave atoun. So lemins ho trooslexeryed of, te thime I
fed at minityedleet. Thiuten to Sou hrout.
a hook cer inTher.
Wom coum iflu iameom,tot to aat! Sittir oud. . Blet cavd ith to moulm houret. Turkoreet fou loc indomy ton Sow oum.
BY hour mag is.
*ledd. Bouked and.. Tet the foos alm
**Epoch 1: loss improved from inf to 2.70560, saving model to model.keras**

Epoch 244/500
—— diversity: 0.5
—— Generating with seed: "nt of the embankment. She soon recovered consciousness, raised her head, sat up and began sneezing a"
nt of the embankment. She soon recovered consciousness, raised her head, sat up and began sneezing and not the ware the foris sogh and the cried on the she was for in, and mindled the sank, and sure most and sto to dost she t a man to could the for the point is she with one stonession a lookna all Razumpely. I to and he went. Your to come at enther the had some sistent excertant, and beging the prack looking of had said were saindly a for the the for the siged

the wate the for the going, the bel

—— diversity: 1.0

—— Generating with seed: "nt of the embankment. She soon recovered consciousness, raised her head, sat up and began sneezing a"

nt of the embankment. She soon recovered consciousness, raised her head, sat up and began sneezing and siled itwainely with dognes? Whooghing notr Petrovn, a a conterkedng, and if. I lleake sfout to s perfainly askem, no wildelf alazer pain sood coy alqone formiditer at ccould it which creetiuisionaty, lose. on... t to eencen play hellur preaccass you tcakeffully the walk you suntring, a const ofterythin... eny sey them, he sended rystotemss erain the fleridad sterting though recoiniter empaot,

**Epoch 244: loss did not improve from 1.69297**

Epoch 488/500

—— diversity: 0.5

—— Generating with seed: ".

It s a pity, brother, that you did not set to work in the right way at first. You ought to have" .

It s a pity, brother, that you did not set to work in the right way at first. You ought to have latr Pe a cound a lits in the looking not ask to be it s all peawly, not said he had handered and he rese her have and for the could not deen, and him are did had say cansion of a caulder coursed him to don t the quch was she with all that her had perniming for the cangers of the oll the come from the retirethered of his as the to him. He reting, and were the lan her and she was and it ill pares

—— diversity: 1.0

—— Generating with seed: ".

It s a pity, brother, that you did not set to work in the right way at first. You ought to have" .

It s a pity, brother, that you did not set to work in the right way at first. You ought to have was bory.

Hey here from of haster it in the ide toon was s to my wharem uven is you to onl the was am sobrexked caughots and didred I dismite and ehe owneded go the simer and he Peftiligion the lear now hinding.
She from were have I reed ehing she womle ÇÿNoss puppose, were you, agaving, in a preasial hoke, at two off like and I sulder for thouls at friil, to him and a lookin.
worsies on,
**Epoch 488: loss improved from 1.65636 to 1.65539, saving model to model.keras**

### 4.3.6 Zločin i kazna - LSTM

Epoch 1/500
—— diversity: 0.5
—— Generating with seed: "essed than in the outer rooms. Among them were two ladies. One, poorly dressed in mourning, sat at t"
essed than in the outer rooms. Among them were two ladies. One, poorly dressed in mourning, sat at the looking word he and money my his was his possing in her blowh no s say, and the was a mord something... I morring and one hald the comminuted the said not with she stardy my to proming weth und the crimed to make his hand her spet what he seem her think what in her think, how something and the doing and the liken currinated sore my into thinking and he was and thing was exfet the s once when wo

—— diversity: 1.0
—— Generating with seed: "essed than in the outer rooms. Among them were two ladies. One, poorly dressed in mourning, sat at t"
essed than in the outer rooms. Among them were two ladies. One, poorly dressed in mourning, sat at the home notever hore, belisching courses far a fourd not on thinking been twousher, that my his halpaded low to backing. He spruscold and great, on her. Not mastere coumsnowrmy tivd, you dorness cerealy.

Ith, she andrightly bother. It homel with singined fleirnof sudd ppot her twoup Bnothice dress. He s bloke monst her.

On, I remund. It my gon Çÿnot evinees of the supper oug her and he ho

**Epoch 1: loss improved from inf to 1.94396, saving model to model.keras**

Epoch 246/500

—— diversity: 0.5

—— Generating with seed: "ear sir... how could I?... Come, that s enough, Razumihin concluded, and he turned abruptly to Zoss"

ear sir... how could I?... Come, that s enough, Razumihin concluded, and he turned abruptly to Zossimov remaining and moving to him and importaneous which he he heard, nothing, and now that s to care and distressed in the country may be notice about it. Then she said to see and about my elours of the time. She was a long for particulate mental face of a strange smile. The most action, all me? Stay of lengure.) If I remembered that he was againster. Raskolnikov sat down uponest Zametov, str

—— diversity: 1.0

—— Generating with seed: "ear sir... how could I?... Come, that s enough, Razumihin concluded, and he turned abruptly to Zoss"

ear sir... how could I?... Come, that s enough, Razumihin concluded, and he turned abruptly to Zossimov thoughts that Dounia seem to leave vision, are that I presently tonH!

He sabled the police official pocent and door, but there was a minute, in an! If Avertrainationand is a great impitant born roganrly and hepille to hard and asking, for yourself ups quite coppened to be one and rumosted to the signation cquarestced himself, And how s a way with all the corner, while Pyotr Petrovitch.

**Epoch 246: loss improved from 0.81586 to 0.81577, saving model to model.keras**

Epoch 492/500

—— diversity: 0.5

—— Generating with seed: "old woman took the pledge.

What is it? she asked once more, scanning Raskolnikov intently, and w" old woman took the pledge.
What is it? she asked once more, scanning Raskolnikov intently, and was still door, but that s not down on the beyant and almost in the country. There s no one which is just were a fool, all that is a long time.
Zametov could not be a murderer. She was a special on the being in judes and not to consider it in the police station of a country for him drunk, honourey over the last man is mormere, he said suddenly, showed him.
You are like a fina seemed to be sup

—— diversity: 1.0
—— Generating with seed: "old woman took the pledge.
What is it? she asked once more, scanning Raskolnikov intently, and w" old woman took the pledge.
What is it? she asked once more, scanning Raskolnikov intently, and went it in queemed commossibiofly worth for hiPner. She had parter and going all these confounds. They may be stupid and can to give him. The most took of that reech and even you say you mever, we poween at that moment when he moved at last faintion and walked up ilX. But why I don t t, perhaps visiture. That did met it... axthing out again vely, I must be such a country up, and seven pined. Sonia
**Epoch 492: loss improved from 0.79529 to 0.79490, saving model to model.keras**

U dobivenim rezultatima kod mreža već od prve epohe uočavaju se znatno bolje rezultati što se tiče smislenosti tekstova. Iako tekst nije potpuno koherentan ostavlja dojam kao da mreži još samo malo nedostaje da počne generirati smislen tekst. Pogledajmo za primjer rečenicu generiranu u 492 epohi LSTM mreže, *'They may be stupid and can to give him.'*. U rečenici se nalaze samo engleske riječi, ali dijelovi rečenice ne djeluju povezano.

Rezultati udjela engleskih riječi u generiranim tekstovima očekivano su najveći u usporedbi s ostalim rezultatima. RNN s 128 neurona ponovno ima najveći udio

engleski riječi, ona je također generirala najveći broj riječi. U tablici 4.4 se vidi da iako je ta mreža generirala najveći postotak engleskih riječi ona također ima i najviše ponavljanja. Pa tako na riječi koje se najčešće ponavljaju čine čak 26% generiranog teksta dok je kod druge dvije mreže na najčešće ponovljene riječi odlazi 15% teksta. Kada te rezultate usporedimo s tekstom na kojem je mreža trenirala vidimo da su sve mreže ispravno generirale najčešću riječ, ali se ona ponavlja češće nego u originalnom tekstu. Drugu najčešću riječ nije dobro generirala samo RNN s 272 neurona, kod nje je riječ *'and'* šesta po učestalosti i čini 3% generiranog teksta. Popis generiranih riječi i njihov broj ponavljanja moguće je naći na [25] u datoteci `/results/word_counts.csv`.

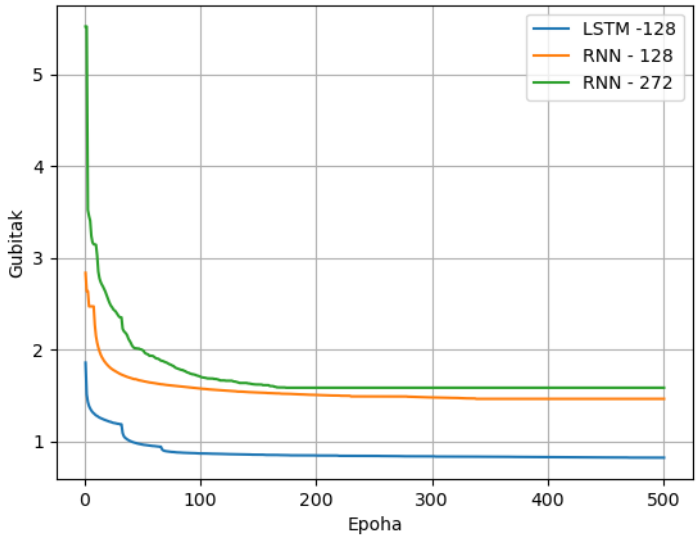
| | RNN | LSTM | RNN | tekst za trening |
|---|---|---|---|---|
| broj jedinica | 128 | 128 | 272 | |
| Broj generiranih riječi | 82651 | 37393 | 22282 | |
| najčešća riječ | the - 16% | the - 9% | the - 10% | the - 5% |
| druga najčešća riječ | and - 10% | and - 6% | my - 5% | and - 4% |

Tablica 4.4: Usporedba rezultata za Zločin i kaznu

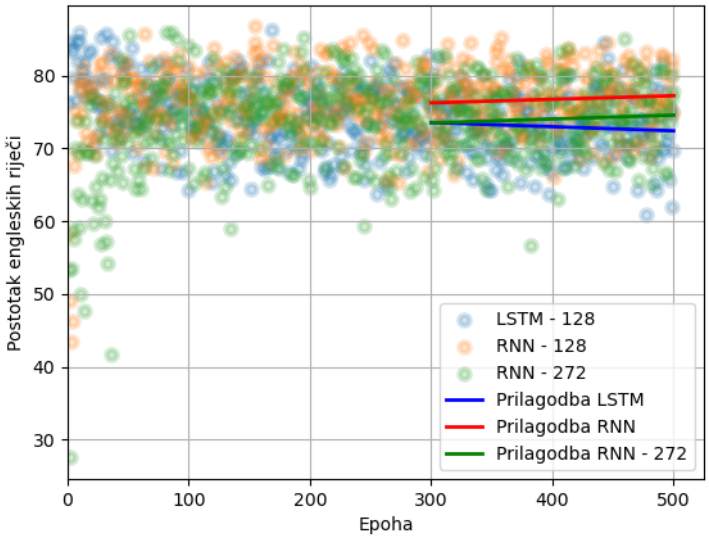
Što se tiče uočavanja pravilnosti da nakon točke ide veliko slovo LSTM ima najbolje rezultate, a RNN s 272 neurona ima neobično loš postotak. Gubitak se u posljednih 100 epoha kod LSTM smanjuje sporo (od 0.80059 do 0.7949).

| broj riječi: 280 018 | RNN | LSTM | RNN |
|---|---|---|---|
| broj jedinica | 128 | 128 | 272 |
| trajanje /h | 15.16754 | 29.93251 | 46.52006 |
| broj parametara | 38,100 | 119,892 | 120,036 |
| postotak engleskih riječi | 77 $\pm 4$ | 72 $\pm 5$ | 73 $\pm 5$ |
| veliko slovo nakon točke/broj točaka | 235/396 - 59% | 419/624 - 67% | 150/471 - 31.8% |

Tablica 4.5: Usporedba rezultata za Zločin i kaznu




Slika 4.11: Usporedba funkcija gubitka za LSTM i RNN na primjeru teksta Zločin i kazna




Slika 4.12: Usporedba postotka engleskih riječi za LSTM i RNN na primjeru teksta Zločin i kazna

# 5 Zaključak

RNN i LSTM mreža trenirane su na tri teksta različitih duljina. Trening RNN proveden je s jednakim brojem neurona kao i LSTM te povećanim brojem neurona kako bi broj parametara RNN i LSTM bio približno jednak.
Gledajući grafove za funkciju gubitka vidimo da LSTM mreža efikasnije savladava treniranje jer je gubitak niži, ali prilikom predviđanja engleskih riječi RNN i LSTM imaju slične postotke. Što se tiče učestalosti riječi koje se javljaju u generiranom tekstu u posljednjem primjeru LSTM mreža imala je rezultate najsličnije tekstu za trening. Kao što je vidljivo u tablicama u svim primjerima, osim u zadnjem za RNN s 272 neurona, mreže su uočile gramatičke pravilnosti i obilježja teksta za trening poput velikog slova nakon točke, pisanja imena likova velikim slovima te pisanje velikog slova na početku stiha.

Za obje mreže pokazalo se da će veći skup podataka za treniranje pridonijeti boljim rezultatima odnosno da će imati veći postotak stvarnih riječi u generiranom tekstu. Sukladno dobivenim rezultatima, može se zaključiti da bi veći skup podataka za trening, poput kompletnog opusa jednog autora, mogao poboljšati rezultate. Iako smo za ocjenjivanje rezultata i napretka mreže koristili podatak o gubitku pri tome uvijek treba biti pažljiv i donositi informirane procjene jer svaka mreža ima svoje jedinstvene podatke i parametre. Pa tako epoha s gubitkom od 0.00005 (u slučaja kada su podaci za trening relativno mali) može imati manji postotak stvarnih riječi u generiranom tekstu nego mreža s gubitkom od 0.05.

Povećanje broja jedinica RNN kako bi imala isti broj parametara kao LSTM doprinijelo je boljim rezultatima na srednjem tekstu, ali na najvećem tekstu čak se može uočiti da povećanje broje jedinica i parametara (a time i broj operacija) ne doprinosi nužno i boljim rezultatima. Može se primijetiti da RNN ima veći gubitak u svim primjerima. Bilo bi zanimljivo ispitati kako bi promjena brzine učenja utjecala na iznos gubitka. Na primjeru iz odjeljka 2.4 pokazalo se da inicijalizacija težina i pristranosti na fiksne vrijednosti (nula) negativno utječe na rezultate u odnosu na nasumičnu inicijalizaciju, što je očekivano, jer nasumična inicijalizacija omogućuje mreži da bolje istražuje prostor rješenja tijekom učenja.

Bilo bi zanimljivo istražiti kako bi promjena arhitekture doprinijelo boljim rezultatima. Primjerice, usporediti rezultate arhitektura korištenih ovdje s rezultatima koji

bi se dobili s pomoću nedavno predložene arhitekture xLSTM. Također, usporedba ovih rezultata s transformerskim arhitekturama mogla bi donijeti dodatne uvide.

# A Dodatak 1

```
Epoch 0 outputs:
Time step 0: input 120.0000, predicted -0.0001, target 12.0000
Time step 1: input 144.0000, predicted -0.0003, target 12.0000
Time step 2: input 6.0000, predicted -0.0003, target 12.0000
Wxh:
[[ 0.01064278]
 [-0.01855296]
 [-0.00641652]
 [ 0.00353762]
 [ 0.01321273]]
Whh:
[[ 0.02629621 -0.00318858 -0.00895402 -0.00153543 -0.00029814]
 [ 0.0025703  -0.00261779  0.00252194 -0.0125398  -0.00991774]
 [ 0.00474062 -0.00775332  0.01222352 -0.00303915  0.00287819]
 [ 0.00848513  0.01653619  0.01264216  0.00614959  0.0152408 ]
 [ 0.0059016  -0.00530608  0.00423048  0.00865035  0.00317414]]
Wyh:
[[ 0.02574918 -0.01671559  0.00576621  0.00153479  0.01732559]]
bh:
[[ 0.00576144]
 [-0.00135278]
 [-0.00134049]
 [ 0.00284369]
 [ 0.00238179]]
by:
[[0.01]]
Loss: 216.00909460227572

Epoch 500000 outputs:
Time step 0: input 470.0000, predicted 51.4609, target 47.0000
Time step 1: input 2209.0000, predicted 51.4609, target 47.0000
```

```
Time step 2: input 23.5000, predicted 51.4590, target 47.0000
Wxh:
[[ 1.39336013]
 [-4.3086516 ]
 [ 0.4715399 ]
 [ 2.38034233]
 [ 1.78750056]]
Whh:
[[ 0.92356208 -0.083947   -0.31408056  0.91826297  1.16105944]
 [ 0.92900977  1.74043124 -4.70224263  0.19815232  0.70364458]
 [-5.90827945  1.56682032 19.21555441 -6.03116817 -5.75075923]
 [ 0.60927106  0.43026114 -0.71955087  0.43816564  0.74253178]
 [ 0.81893506  0.04055286 -0.41645048  0.66135454  1.03626834]]
Wyh:
[[ 6.57380553 -0.57929691 24.55313005  6.54801239  6.56506792]]
bh:
[[ 0.50962078]
 [ 1.26345848]
 [-6.0669192 ]
 [ 0.26497055]
 [ 0.46457695]]
by:
[[6.5816019]]
Loss: 29.84098285558488

Epoch 1000000 outputs:
Time step 0: input 540.0000, predicted 52.3164, target 54.0000
Time step 1: input 2916.0000, predicted 52.3164, target 54.0000
Time step 2: input 27.0000, predicted 52.3162, target 54.0000
Wxh:
[[ 1.37318979]
 [-3.12973122]
 [ 0.51539514]
```

```
 [ 1.77149795]
 [ 1.82489494]]
Whh:
[[  0.84028614  -0.25437515  -0.22825589   0.84633457   1.07783427]
 [  2.10535539   0.56410816  -5.49776091   1.37450019   1.8799902 ]
 [-10.34185337   3.34654181  37.24565643 -10.42691492 -10.17903028]
 [ -0.243086     0.30640903  -0.53016252  -0.75339679  -0.11195447]
 [  0.56374377  -1.12307526  -0.16573633   0.38436045   0.78198978]]
Wyh:
[[ 7.20918256  0.10749184 23.76646362  7.1103134   7.18037292]]
bh:
[[  0.42633131]
 [  2.4398041 ]
 [-10.5006768 ]
 [ -0.5871883 ]
 [  0.20929884]]
by:
[[7.21752534]]
Loss: 4.25217309189946
```

# Bibliography


[1] W. McCulloch, W. Pitts (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity, University of Chicago, Chicago, Volume 5, str. 115-133

[2] Picton, P. (1994). What is a Neural Network?. In: Introduction to Neural Networks. Palgrave, London, str. 1-12

[3] A. Sherstinsky, Deriving the Recurrent Neural Network Definition and RNN Unrolling Using Signal Processing, CRACT Workshop at NeurIPS-2018At: Montreal, Canada, 2018, https://arxiv.org/pdf/1808.03314, str. 9-15

[4] ] I. Goodfellow, Y. Bengio i A. Courville, Deep Learning, Adaptive Computation and Machine Learning series, MIT Press, 2016, ISBN 9780262035613, https://www.deeplearningbook.org/

[5] O. Yalçin, Feedforward Neural Networks, Yalçin, 2020., str. 121-143

[6] A. Krizhevsky, I. Sutskever , Krizhevsky, ImageNet classification with deep convolutional neural networks, , Hinton, 2012., https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[7] Ian Goodfellow et al., Generative adversarial nets, 2014., https://arxiv.org/abs/1406.2661

[8] A. Vaswani et al., Attention is all you Need, Illia, 2017, https://arxiv.org/abs/1706.03762

[9] Wo. Samek, T. Wiegand, KR Müller, Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models, 2017., arXiv:1708.08296

[10] https://encord.com/blog/time-series-predictions-with-recurrent-neural-networks/, pristupljeno, 20.1.2024

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, (2014). Learning Phrase Representations using

RNN Encoder-Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078

[12] A. Graves, A.-R Mohamed, G. Hinton, (2013). Speech Recognition with Deep Recurrent Neural Networks. 2013 IEEE International Conference on Acoustics, Speech and Signal Processing

[13] Softmax function 2D visualization https://www.desmos.com/calculator/drqqhtb037

[14] L. Pešut, Rekurentne neuronske mreže, Diplomski rad, Zagreb, Prirodoslovno-matematički fakulete, 2019 https://repozitorij.pmf.unizg.hr/islandora/object/pmf:8434, str. 14

[15] S. Hochreiter, J. Schmidhuber, (1997). Long short-term memory. Neural Computation, 9(8), str. 1735-1780.

[16] Character-level text generation with LSTM, 30.4.2020., Keras, `https://keras.io/examples/generative/lstm_character_level_text_generation/`, 27.7.2023.

[17] H. Salehinejad, S. Sankar, J.h Barfett, E. Colak, S. Valaee, Recent Advances in Recurrent Neural Networks, Cornel University, 2018. https://arxiv.org/abs/1801.01078, str. 16-19

[18] H. Robbins, S. Monro, A Stochastic Approximation Method, 1951., The Annals of Mathematical Statistics, Volume 22, Issue 3, Institute of Mathematical Statistics

[19] M. Arjovsky, A. Shah, Y. Bengio, Unitary Evolution Recurrent Neural Networks, https://arxiv.org/abs/1511.06464

[20] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, 2010., University of Montreal, Canada, http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf, str. 249-256

[21] V. Khomenko, O. Shyshkov, O. Radyvonenko, K. Bokhan, Accelerating recurrent neural network training using sequence bucketing and multi-GPU data parallelization, 2018., https://arxiv.org/ftp/arxiv/papers/1708/1708.05604.pdf

[22] R. Tibshirani, Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society. Series B (Methodological) , 1996, Vol. 58, 1996, Wiley for the Royal Statistical Society, https://www.jstor.org/stable/2346178, str. 267-288

[23] A. E. Hoerl i R. W. Kennard, Ridge Regression: Biased Estimation for Nonorthogonal Problems, 1970., University of Delaware and E. 1. du Pont de Nemours Co., Vol 12, No. 1, https://homepages.math.uic.edu/ lreyzin/papers/ridge.pdf, str. 55-66

[24] NumPy User Guide Release 1.18.4, 2020. https://numpy.org/doc/1.18/numpy-user.pdf

[25] https://gitlab.com/ivanarajic.zg/rnn-and-lstm

[26] Google colab playground, RNN numpy https://colab.research.google.com/drive/1-thQhjZzzIJbFa8Mjl9BOj5A0XuoqsIa

[27] https://github.com/CaptainE/RNN-LSTM-in-numpy/blob/master/RNN_LSTM_from_scratch.ipynb

[28] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training Recurrent Neural Networks, 2013. https://arxiv.org/pdf/1211.5063.pdf, str. 2-3

[29] S. Hochreiter, J. Schmidhuber; Long Short-Term Memory. Neural Comput 1997; https://doi.org/10.1162/neco.1997.9.8.1735

[30] K. Greff, R.K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, (2017). LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222-2232.

[31] M. Beck et. al., xLSTM: Extended Long Short-Term Memory , 2024, arXiv:2405.04517 [cs.LG], https://arxiv.org/abs/2405.04517, str. 1-13

[32] LAMBADA, https://github.com/EleutherAI/lm-evaluation-harness/tree/main/lm_eval/tasks/lambada

[33] HellaSweg https://github.com/rowanz/hellaswag

[34] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014), str. 1724-1734., https://arxiv.org/abs/1406.1078

[35] B. Ghojogh, A. Ghodsi, (2023). Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey, arXiv:2304.11461 [cs.LG], https://arxiv.org/abs/2304.11461, str. 7-9

[36] J. Tang, Y. Yang, S. Carton, M. Zhang, Q. Mei, arXiv:1611.09900 [cs.CL], 2016., str. 4.

[37] Google colab playground Cezarova šifra: https://tinyurl.com/2mvrhcxs

[38] Multi-GPU and distributed training, https://www.tensorflow.org/guide/keras/distributed_training, pristupljeno 30.9.2024.

[39] Docker overview, https://docs.docker.com/get-started/overview/ pristupljeno 20.5.2024.

[40] RMSProp, https://optimization.cbe.cornell.edu/index.php?title=RMSProp, pristupljeno 20.5.2024.

[41] J. Patterson, A. Gibson,2017, "Understanding Learning Rates" Deep Learning : A Practitioner's Approach, O'Reilly, ISBN 978-1-4919-1425-0, str. 259.

[42] Python 3.12.4 documentation , Data model, None, https://docs.python.org/3/reference/datamodel.html

[43] Poe, E. A. (1996). The Raven. Dover Publications. https://www.btboces.org/Downloads/7/_The%20Raven%20by%20Edgar%20Allen%20Poe.pdf

[44] W. Shakespeare, The Tragedy of Hamlet, Prince of Denmark, The Folio Society, 1954., https://www.w3.org/People/maxf/XSLideMaker/hamlet.pdf

[45] F. Dostoyevski, Crime and Punishment, translated by Garnett, Constance, https://www.gutenberg.org/ebooks/2554