Generiranje teksta pomoću RNN mreža

Marin Benčević

Projektni zadatak - Raspoznavanje uzoraka i strojno učenje

FERIT, Osijek

14. 2. 2019.

Uvod

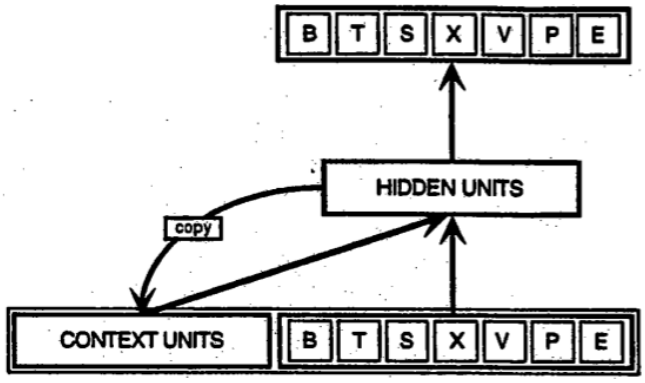
RNN mreže (*eng. recurrent neural network*) su tip neuronskih mreža čije su značajke međusobno povezane u matematički graf. Tipična feed-forward neuronska mreža očekuje fiksnu veličinu inputa i rezultat predikcije je fiksna veličina outputa. S druge strane, RNN mreže rade s nizovima podataka. Također, obzirom da RNN mreže imaju strukturu nalik grafa, njihovo se ponašanje može mijenjati kroz vrijeme ovisno o tome koje su elemente niza dobile do sada. Drugim riječina, RNN može imati interno stanje na koje mreža utječe, a koje se mijenja s ulaznim podacima. To ih čini pogodnim za prepoznavanje govora, rukopisa i, ono što će se obrađivati u ovom radu, modeliranje jezika.

U ovom će radu biti predstavljene RNN mreže i njihov osnovni princip rada. Također, u ovom će se radu trenirati RNN neuronska mreža u svrhu generiranja novih nizova tekstova. Trenirat će se RNN mreža na djelu Judita Marka Marulića. Generirat će se različiti novi primjeri tekstova koji ne postoje u izvornom djelu.

Generiranje teksta pomoću RNN mreža

Princip rada RNN mreže

Glavna razlika RNN mreže i feed-forward mreže je povratna veza. U feed-forward mreži podaci podaci kroz neurone putuju u jednom smjeru, prema izlaznom sloju. U RNN mreži, podaci se pomoću povratne veze mogu vratiti u prijašnji sloj i zajedno sa novim ulazom ponovno proći kroz mrežu.



Dakle, RNN mreže kao ulaz imaju podatke, ali i informacije o podacima koje su dobile u prijašnjim iteracijama. Možemo reći da RNN mreža “sprema” informacije o prijašnjim podacima, tako što izlaze nekih slojeva vraća na ulaz u mrežu. To znači da se predikcija RNN mreže mijenja s brojem predivđanja, jer sa svakim novim predviđanjem ima više dostupnih informacija. Podaci koje je RNN mreža vidjela u prošlosti utjecat će na predviđanje za trenutni ulaz. Možemo reći da RNN mreža, osim korelacije u ulaznim podacima, također pronalazi vremenske korelacije između dva podatka. To ih čini pogodnim za analiziranje sljedova povezanih podataka, kao što su tekst i govor. Ako osoba kaže “gladan sam”, vjerojatnije je da će iduća rečenica biti “idemo jesti” nego “idemo u kino”. RNN mreža pogodna je za pronalazak takvih veza.

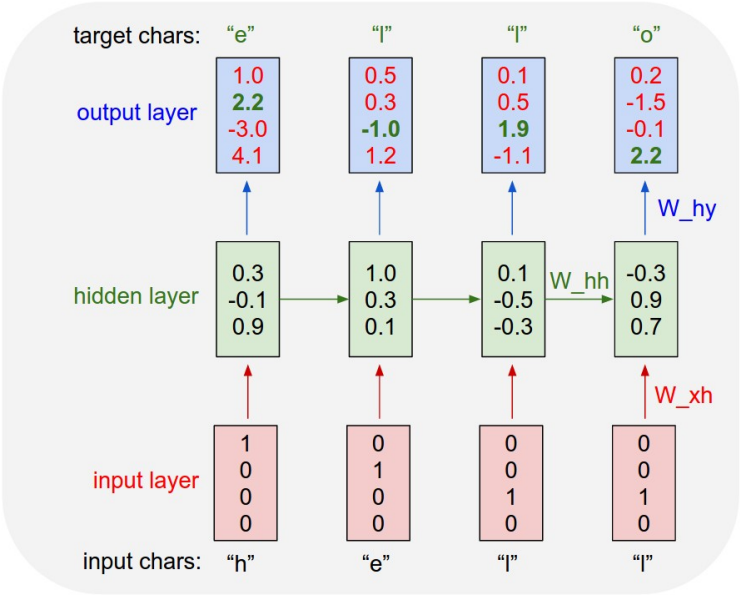
Generiranje teksta pomoću RNN mreža

Matematički, informacije o prijašnjim podacima RNN mreže (tzv. skriveno stanje) u nekom vremenskom koraku t možemo prikazati sljedećom formulom:

$$\mathbf{h}_t = \phi\left(W\mathbf{x}_t + U\mathbf{h}_{t-1}\right),$$

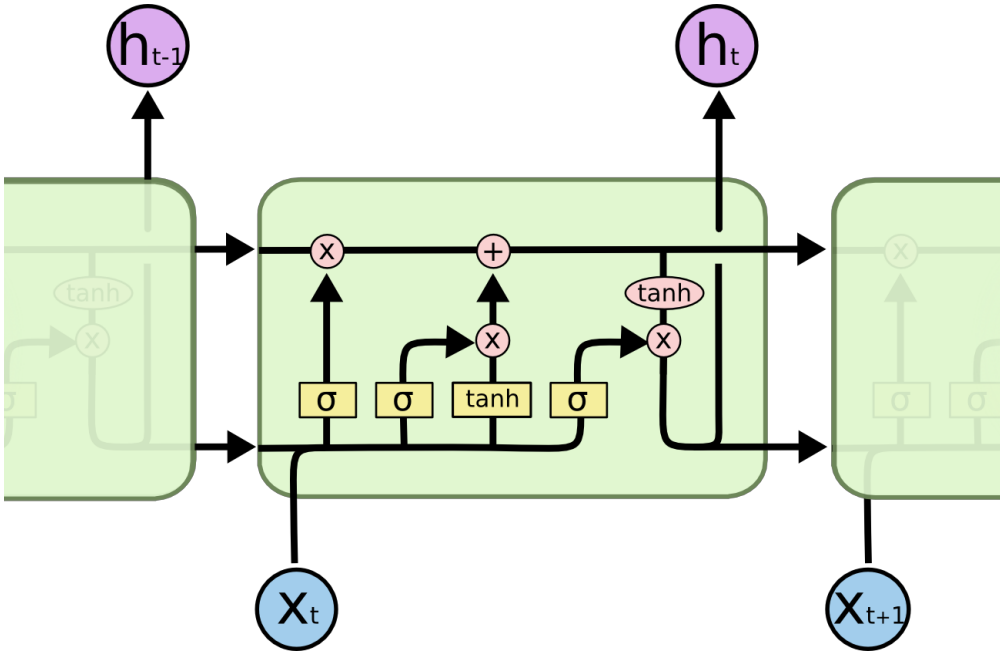
Gdje je *t* vremenski korak, *W* je matrica težina za ulazne podatke, *xt* ulazni podaci za vremenski korak *t*, *U* matrica težina za skriveno stanje. Iz formule vidimo da se skriveno stanje u svakom koraku promijeni ovisno o novim ulaznim podacima.

Izlaz is mreže je tada skalarni produkt skrivenog stanja *ht* i matrice težina *Z* koja je drukčija od matrica *W* i *U*. Odnosno možemo reći da RNN mreže posjeduju lančanu strukturu za niz podataka jer na ulaz u jednu jedinicu uz novi podatak dolaze i informacije o dosadašnjim ulazima u mrežu.STM



LSTM (engl. *long short-term memory*) mreže su jedne od mogućih jedinica RNN mreže. Prednost LSTM jedinica je što mogu pamtiti podatke kroz velik broj iteracija, tako da svaki podatak dugoročno utječe na rezultat. LSTM to postiže izravnim slanjem stanja iz jedne jedinice u drugu. Kao i kod drugih RNN jedinica, svaka LSTM jedinica

ima za ulaz novi podatak kao i stanje i izlaz iz prošle jedinice, a svoj izlaz i novo stanje prosljeđuje i u sljedeću jedinicu, time postižući lančanu strukturu.



LSTM stanje iz jedne jedinice prenosi u drugu, ali putem odlučuje koji dijelovi stanja su bitni i kako će promijeniti stanje ovisno o novom ulazu. To radi pomoću troja tzv. “vrata”, koja se ili skalarno množe ili zbrajaju sa stanjem iz prijašnje LSTM jedinice, i tako ga mijenjaju. To se novo stanje onda prosljeđuje u iduću jedinicu, zajedno sa izlazom iz te jedinice.

Prva vrata su vrata zaboravljanja (*engl. “forget layer gate”*). Ovise o produktu izlaza iz prijašnje jedinice i novog podatka, koji prolazi kroz sigmoid funkciju i onda se skalarno množi sa dosadašnjim stanjem. To množenje omogućuje prigušavanje određenih dijelova ulaznog stanja, odnosno “zaboravljanje” stanja. Množenje s jedinicom zadržava taj dio stanja, dok množenje s nulom potpuno “zaboravi” taj dio stanja, tako da ne utječe na daljnji izlaz.

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

Druga su vrata ulazna vrata (engl. *“input gate layer”*). Ova vrata određuju koji dijelovi stanja će se promijeniti ovisno o novom ulaznom podatku. Produkt novog ulaza i prošlog izlaza prolazi kroz sigmoid sloj, kao u prvim vratima. Ako je izlaz nula, taj se dio stanja neće promijeniti.

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

Treća vrata određuju samu promjenu stanja množenjem tog istog produkta sa *tanh* slojem. Time se kreiraju promjene stanja za svaki element stanja.

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

Rezultat drugih i trećih vrata se međusobno množi (tako da se promijene samo određeni dijelovi stanja), i taj se produkt pribraja na dosadašnje stanje.

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

Nakon što je promijenjeno stanje, ono se prosljeđuje u sljedeću jedinicu. Također, to stanje prolazi kroz *tanh* sloj, i množi se s izlazom iz sigmoid sloja ulaza. Taj produkt određuje izlaz iz LSTM jedinice. Odnosno, izlaz ovisi o novom stanju kojeg je LSTM jedinica kreirala i o ulazu u jedinicu. Taj se izlaz također prosljeđuje u sljedeću jedinicu.

Vidljivo je kako LSTM može podatke iz prvog ulaza prosljeđivati sve do samog kraja mreže, time postižući dugotrajno “pamćenje”. Na svakom koraku, LSTM odlučuje koje će dijelove stanja zaboraviti i koje će dijelove stanja promijeniti ovisno o novom ulazu. Time se mogu iskazati kompleksne vremenske korelacije među podacima.

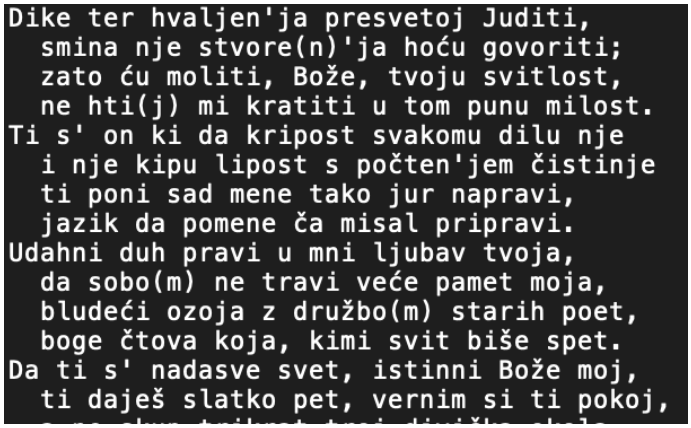
Generiranje teksta pomoću RNN mreža

Implementacija RNN mreže

Implementirane je mreža trenirana na razini slova na djelu Judita Marka Marulića.

Generiranje Judite

Podatkovni skup za generiranje Judite je cijelo djelo Judita, bez sadržaja, naslova, predgovora i sl. tako da su jedini podaci strofe djela. Ukupno podatkovni skup sadrži 81.119 slova, od kojih su 66 unikatnih slova koji čine vokabular RNN mreže.



Ulazni podaci za mrežu bit će nizovi slova, dok će iznazni podaci biti sljedeće slovo ulaznog niza. Npr. jedan ulazni podatak je “Dike te”, dok je odgovarajući izlaz “r”. Ulaze i izlazi dobiju se iteriranjem kroz cijelo djelo svaka 3 slova. Pri svakoj iteraciji, spremaju se sljedećih 39 slova od sadašnjeg slova u ulazne podatke, a 40. slovo u izlazne. Na djelu Judita dobije se oko 27.000 podataka.

```
part_len = 40
step = 3

parts = []
next_chars = []

for i in range(0, len(sample) - part_len, step):
    # Podijeli tekst na niz dijelova koji se preklapaju, svakih `step` slova
    # npr. "Bok, ja sam Marin" ", "ja sam Marin Ben", "sam Marin Bencev" ...
    parts.append(sample[i : i + part_len])
    # i za svaki taj dio spremi u `next_parts` sljedece slovo recenice
    # npr. ""
    next_chars.append(sample[i + part_len])
```

Generiranje teksta pomoću RNN mreža

U podacima se svaka riječ kodira kao boolean vektor kojem je dužina jednaka broju jedinstvenih slova, a svaka vrijednost je False, osim vrijednosti koja se nalazi na indeksu te riječi u listi unikatnih slova. Time smo postigli one-hot encoding za podatke.

```
# Kodiraj input i output. Za svako slovo napravi niz boolova takav da su
# sve vrijednosti false osim vrijednosti na indeksu tog slova.
# Za output koristi indekse od next_chars. Tako da za svaki input
# (koji je niz indeksa), output bude indeks sljedeceg slova.

x = np.zeros((len(parts), part_len, len(unique_chars)), dtype=np.bool)
y = np.zeros((len(parts), len(unique_chars)), dtype=np.bool)
for i, part in enumerate(parts):
    for j, char in enumerate(part):
        x[i, j, char_to_index[char]] = 1
    y[i, char_to_index[next_chars[i]]] = 1
```

Model kojeg koristimo sastoji se od 2 LSTM sloja od 265 jedinica, sa dropout slojem između njih. Nakon toga slijedi jedan potpuno povezani sloj sa softmax aktivacijom kako bi se dobila vjerojatnost za svaku riječ.

```
model = Sequential()
model.add(LSTM(256, input_shape=(part_len, len(unique_chars)), return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(256, input_shape=(part_len, len(unique_chars))))
model.add(Dense(len(unique_chars), activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam')
```

Model se trenira na svim podacima, sa veličinom batcheva 128 i u 50 epoha. Koristi se checkpointing kako bismo mogli spremiti model na epohi s najmanjom greškom.

Nakon što se istrenira model, da bismo mogli generirati tekst moramo koristiti model za predviđanje slijedećeg slova. Počet ćemo od nasumičnog dijela ulaza, i pred-

vidjeti sljedeće slovo. Nakon što imamo to slovo, ispisat ćemo ga i dodati na ulaz, a sa ulaza ukloniti prvo slovo. Time ćemo generirati slovo po slovo.

```
start = random.randint(0, len(x) - 1)
generated = ''
part = sample[start : start + part_len]
generated += part

for i in range(1000):
    x_pred = np.zeros((1, part_len, len(unique_chars)))
    for j, char in enumerate(part):
        x_pred[0, j, char_to_index[char]] = 1.

    preds = model.predict(x_pred, verbose=0)[0]
    next_index = sample_preds(preds, 0.8)
    next_char = index_to_char[next_index]

    generated += next_char
    part = part[1:] + next_char

    sys.stdout.write(next_char)
    sys.stdout.flush()

print()
```

Za razliku od ostalih tipova predikcije, rezultati RNN mreže mogu biti znantno bolji ako se kod previđanja ne koristi najvjerojatniji rezultat. Ako se uvijek uzima najvjerojatnije sljedeće slovo, u tekstu ima puno manje varijacija i određene fraze i riječi počinju se ponavljati. Zato je potrebno dodati element nasumičnosti u predviđanja.

```
# Za listu predvidjenih indeksa slova odaberi jedno slovo.
# `temperature` odredjuje moguce odstupanje od najbolje
# vrijednosti.
def sample_preds(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

Generiranje teksta pomoću RNN mreža

Nakon svakog predviđanja, dobrim rezultatima ćemo smanjiti vjerojatnost, a lošim rezultatima povećati vjerojatnost za faktor koji ovisi o konstnati koju nazivamo temperatura. Od tih novih rezultata napravit ćemo multinomnu statističku distribuciju i uzeti nasumičan rezultat. Ako je temperatura viša, “dobri” rezultati će imati manju relativnu vjerojatnost, i veća će vjerojatnost bit za odabir “lošijih” rezultata.

U nastavku slijede primjeri rezultata koje dobijemo iz modela:

```
O misi sm pastilu kriza nese u gori,
  a ne konja ti sten zlača te prede mnidi
  i pri(j)a za njim poparva gu pukiti;
  dostoje neseći: Da te pez ukvarist.
Ni bi je zavaše sert si budi praviti
  i parazvaru sarbi pukali gavoriti
  ne s tubi(j)aše ki po da napodi!
  jer ju bi sidaje razskki, viši pamo,
  kako ki da ne hode, kin slaga vladu
  gdi loviv: ne kar na poglepu privini,
  i lapad prižjnima da pridavšene pom.
  Vičar nji posom m ljubi prilavami
mogam.
```

```
Ka bi poz pripiše u znam ti priditi,
  da ga nje pokoga, vaz nas oboju ki.
  Bog i ka ne ostaju za mno tere mnoga,
  ni se ne bit zama li stog zazča se pori?
Sipo na pogu plavci pohoće,
  u tvara, da zaman svita bi holita;
  Mazača svakahu ku ji postebaše,
  oliža, kako ta povarča gaviri.
  Kark Ejam svitlust svar bud, i stoju dum.
```

Iz primjera je vidljivo kako je mreža oprilike naučila dobru dužinu stihova. Također je naučila da se strofe sastoje od prvog stiha, a svaki sljedeći stih u strofi je uvučen. Mreža također koristi interpunkcijske znakove na dobrim mjestima (kraj stiha) i čak je naučila suglasnike u zagradama koji se pojavljuju u prijevodu na suvremeni jezik. (npr. “ubi(j)aše”)

S druge strane, mreža većinom daje krivu duljinu strofe (u originalu je uvijek četiri stiha) i riječi koje koristi su često nepostojeće riječi, čak i na rensesansnoj splitskoj čakavštini. Stihovi većinom nemaju smisla i nisu povezani jedni s drugima.

Generiranje teksta pomoću RNN mreža

Kad uzmemo prednosti i nedostatke u obzir, obzirom na mali broj podataka (samo jedno djelo) i na to da je mreža učena od nule, bez ikakvog predznanja o jeziku ili strukturi, mreža dobro oponaša Marulića. RNN mreže nisu dovoljno dobre da bi generirale potpuno smislen tekst, čak i na većim podatkovnim skupovima.

Mijenjanjem temperature mijenjaju se i rezultati. U nastavku slijede rezultati za različite iznose temperature.

temp = 0.5	temp = 0.8
Toj strose opriti, same zaspodit. Zlubavi moći pom,, virti bilu puliti, jer glavu, timi svitu da je je reza ner svita zapilu uzenitu, s tveju. Čada sva da po stih nesobosi svahu, u plavom mojuh bih kuše po vemom hvoje, gdi se naš njih potaj prost	Ni moći, prigniti dokor uzihciti, s trubim poloče, nimdi gdi (j)e pili, svit od poslišuti, da se modu razsi. Svako se pretazsve da stagu na svori, njugu ti pred toj mor starti vrabla nastov. Tu samsti vakasa svarži ki pritil, i naš ubih pohliti, alima si slogoš.
temp = 1.0	temp = 1.2
j pataj prostvol neskori, ne bih gonji zlabom poklopu tij kako ni ci smanje čajpo meća bim. Paka ju da pri rumo litarće da svita, za svit za priduje da ponom pohuće, kad priča njegavi ovol ne priti, s prevavim zlobda Jer si Jerosili i smata gradpvi na , nece se potiti.	ina tajme ozvajaliti svituji, ca ploštav zlatiše prosuti, naprad, čarvi se on kralje od Bogu, izide narubi svit postaju nogor. Jude Zahul')aro se prativiti zato se veslaši, li da nje prikučti, svu reli, pistita, ne ker od njeuči. I s nimim gospori od tica či se mone.

Za niske temperature mreža velikom većinom generira stvarne riječi koje se mogu pronaći u tekstu i prati izvornu strukturu (4 kitice po strofi), ali nekad ponavlja slova ili riječi (npr. “, ,” ili “je je”). Za više temperature mreža bira manje vjerojatna slova, tako da često stvara nove riječi koje u izvornom tekstu ne postoje (“prevavim zlobda Jer si Jerosili”), koristi velika slova, interpunkciju i razmake na krivim mjestima.

Generiranje teksta pomoću RNN mreža

Evaluacija modela i zaključak

Model je kroz 50 epoha pao na grešku od 0.09, dok je nakon 1. epohe greška bila 3.2. Međutim, za generiranje teksta teško je interpretirati standarde metrike koje se koriste za strojno učenje. Najbolji pokazatelj kvalitete generiranog teksta je mogućnost raspoznavanja generiranih tekstova od tekstova iz podatkovnog skupa. U tu svrhu kreirana je anketa gdje su ispitanicima nasumičnim redoslijedom prikazane izvorne strofe Judite i strofe koje je generirala mreža. Za svaku strofu ispitanici su morali odabrati “Marulić” ako misle da je to izvorni tekst, “Robot” ako misle da je automatski generirano i “Nisam siguran/a” ako nisu sigurni.

U anketi je bilo 7 različitih strofa, od kojih je 3 bilo iz Judite, a 4 koje su automatski generirane. Za izvorne tekstove, oko 80.93% ispitanika je točno prepoznalo da se radi o izvornom tekstu. Njih 9.53% nije bilo sigurno, dok su ostalih 9.53% odgovorili netočno, odnosno mislili su da je tekst automatski generiran. S druge strane, generirani tekstovi su u 82,13% slučajeva točno prepoznati. Ispitanici u 7,15% slučajeva nisu bili sigurni, dok su u 10,73% slučajeva mislili da se radi o izvornom tekstu.

Iz rezultata se vidi da su u više od 10% slučajeva ispitanici “zavarani”, odnosno mislili su da je generirani tekst stvaran. Međutim, u više od 9% slučajeva je Marulić uspio zavarati ispitanike tako da misle da je njegov izvorni tekst automatski generiran. Razlika između pogrešnog prepoznavanja Marulića i generiranih tekstova je samo 1,2%, tako da generirani tekstovi nisu vrlo vjerodostojni ori ginalu.

Generiranje teksta pomoću RNN mreža

Najveća razlika originala i generiranih tekstova je šro original ima ritam i rimu, dok generirani tekstovi nemaju. Također generirani tekst proizvodi rečenice koje nemaju smisla, iako koriste stvarne riječi i rečeničnu strukturu.

Obzirom na malu veličinu podatkovnog skupa (samo jedno djelo), iznenađujuće je koliko je RNN efektivno naučila reproducirati tekst.

Generiranje teksta pomoću RNN mreža

Literatura

Andrej Karpathy, The Unreasonable Effectiveness of Recurrent Neural Networks

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Alex Graves, Generating Sequences With Recurrent Neural Networks

arXiv:1308.0850 [cs.NE] https://arxiv.org/abs/1308.0850

Ilya Sutskever, James Martens, Geoffrey Hinton, Generating Text with Recurrent Neural Networks

http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf

Andrej Karpathy, char-rnn Project

https://github.com/karpathy/char-rnn

Sepp Hochreiter, Jurgen Schmidhuber et al, Long Short Term Memory

http://www.bioinf.jku.at/publications/older/2604.pdf

Christopher Olah, Understanding LSTM Networks

http://colah.github.io/posts/2015-08-Understanding-LSTMs/