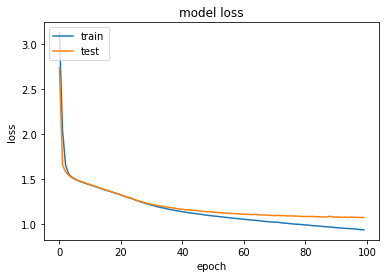
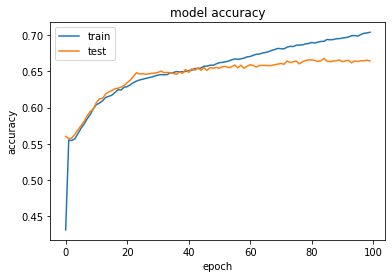
Atanas Delevski

ECE 559 Homework #5 Report

12/01/2020

Part 2):





I decided to use categorical cross entropy as my loss function. I was able to just call it using the Keras library when creating my sequential model. Keras then calculates the cross-entropy loss between the labels and predictions, or in this case between our Xi and Yi. Categorical cross-entropy loss lets us structure our model in such a way that we can then use the softmax function to decide which output to pick from our output vector.

My loss function ended up not going completely to zero, but instead converging to around 1. I think this is due to the fact that my network did not overfit the data, and so therefore it did not get rid of loss completely. I think maybe this was because I decided to have a validation set split, which helped the network learn. I also made sure to use call-backs. This ensured that if the network saw an increase in loss, it would go back and reload the most recent weights so that the loss was always going down.

Part 3)

For my design choice for which letters to pick, I simply chose randomly between the top three letter probabilities. Instead of making it a higher percentage to choose the highest likelihood letter, and a lower percentage for the second highest likelihood letter, etc… I just simply randomly chose from the top three. This seemed to work alright, but then I ended up getting some repetitions of names. To counteract this, I added a while loop to check if a certain name had been generated already, and if so, it would not count it as one of the “20” that it was trying to generate.

Names:

***“A” Names:***

Ania

Anda

Arely

Ally

Ande

Anne

Anily

Anna

Alle

Arali

Arie

Alen

Anie

Anila

Aley

Arand

Arey

Arala

Aray

Arani

X names on next page

***“X” Names:***

Xon

Xolia

Xila

Xan

Xis

Xin

Xally

Xallie

Xale

Xalia

Xola

Xilla

Xor

Xar

Xalin

Xallee

Xile

Xole

Xolin

Xillia

Code on next page

Code:

import keras

from keras import layers

import numpy as np

import random

import io

from keras.models import Sequential

from keras.layers import Dense, Dropout, LSTM

from keras.utils import np\_utils

from keras.callbacks import ModelCheckpoint

from matplotlib import pyplot as plt

import tensorflow as tf

########## PARAMETERS ###############

seq\_length = 11

EON = '~'

input\_letter = 'a' ## <-------------- THIS IS WHERE YOU CHOOSE WHAT LETTER TO INPUT

#####################################

########## PRE-PROCESSING ##############

with open('names.txt', 'r') as names:

lines = [line.rstrip('\n') for line in names]

new\_names = open("new\_names.txt", "w")

for line in lines:

new\_line = line.lower()

while len(new\_line) < seq\_length:

new\_line += EON

new\_names.write(f"{new\_line}\n")

names.close()

new\_names.close()

##########################################

with io.open("new\_names.txt", "r") as f:

text = f.read().lower()

text = text.replace("\n", "")

print("Total length:", len(text))

chars = sorted(list(set(text)))

chars = chars[-1:] + chars[:-1]

print("Total chars:", len(chars))

char\_indices = dict((c, i) for i, c in enumerate(chars))

indices\_char = dict((i, c) for i, c in enumerate(chars))

maxlen = seq\_length

names = []

for i in range(0, len(text), maxlen):

names.append(text[i : i + maxlen])

print("Number of sequences:", len(names))

x = np.zeros((len(names), maxlen, len(chars)), dtype=np.int)

y = np.zeros((len(names), maxlen, len(chars)), dtype=np.int)

eon = np.zeros((len(chars)))

eon[0] = 1

for i, name in enumerate(names):

for t, char in enumerate(name):

x[i, t, char\_indices[char]] = 1

y[i][:-1] = x[i][1:]

y[i][-1] = eon

model = Sequential()

model.add(LSTM(128, input\_shape=(maxlen, len(chars)), return\_sequences=True))

model.add(Dense(len(chars), activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

filepath = "model.hdf5"

checkpoint = ModelCheckpoint(filepath, monitor='loss', verbose=1, save\_best\_only=True, mode='min')

desired\_callbacks = [checkpoint]

# for i in range(epochs):

history = model.fit(x, y, epochs=100, validation\_split=0.15, batch\_size=128, callbacks=desired\_callbacks)

model.save('my\_model.hdf5')

# list all data in history

print(history.history.keys())

# summarize history for accuracy

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

# summarize history for loss

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

model = tf.keras.models.load\_model('my\_model.hdf5')

names\_list = []

while len(names\_list) < 20:

sequence = input\_letter

x\_pred = np.zeros((1, maxlen, len(chars)))

for i in range(maxlen):

for t, char in enumerate(sequence):

x\_pred[0, t, char\_indices[char]] = 1.0

preds = model.predict(x\_pred, verbose=0)

indices = np.argsort(-1\*preds[0][i])[:3]

if 0 in indices:

sequence += EON

else:

sequence += indices\_char[random.choice(indices)]

if sequence not in names\_list:

names\_list.append(sequence)

for name in names\_list:

for char in name:

if char == '~':

index = name.index(char)

neat = name[:index]

print(neat.title())