Text

Generation

Using LSTM Model

{ Long short term memory network : a type of Recurrent Neural Network }

Created by:

Aditi Kumar - 706/IT/14

Arushi Bhatt- 718/IT/14

Dhruv Kameshwar-730/IT/14

ABSTRACT

Neural networks which possess the capability to learn by themselves without any interference was first introduced in 1950's since then have advanced a lot with coming of single layer feed forward to multilayer feed forward then recurrent neural networks and so on . Here we have made use of the LSTM (Long short term memory network), which is a type of RNN and also possess the capability of remembering the information for a longer duration of time .

Recurrent neural networks can also be used as generative models. This means that in addition to being used for predictive models (making predictions) they can learn the sequences of a problem and then generate entirely new plausible sequences for the problem domain. Generative models like this are useful not only to study how well a model has learned a problem, but to learn more about the problem domain itself.

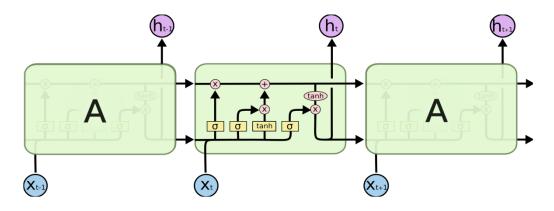
In this project we discovered how to create a generative model for text, character-by-character using LSTM recurrent neural networks in Python with Keras.

THEORY:

A recurrent neural network (RNN) is a class of <u>artificial neural network</u> where connections between units form a <u>directed cycle</u>. This allows it to exhibit dynamic temporal behavior. Unlike <u>feedforward neural networks</u>, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented, connected <u>handwriting recognition</u> or speech recognition.

Long short-term memory (LSTM) is a <u>recurrent neural network</u> (RNN) architecture that remembers values over arbitrary intervals. Stored values are not modified as learning proceeds[<u>further explanation needed</u>]. RNNs[<u>clarification needed</u>]allow forward and backward connections between neurons.

An LSTM is well-suited to <u>classify</u>, <u>process</u> and <u>predict time series</u> given <u>time lags</u> of unknown size and duration between important events. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs[<u>examples needed</u>], <u>hidden Markov models</u> and other sequence learning methods in numerous applications.



The repeating module in an LSTM contains four interacting layers.

INPUT:

- Dataset is the whole novel of 'Alice's Adventures in Wonderland by Lewis Carroll'.
- Dataset was <u>downloaded as complete text in ASCII format</u> (Plain Text UTF-8) placed it in your working directory with the filename **wonderland.txt.**

- 🗆 X

File Edit Format View Help

ALICE'S ADVENTURES IN WONDERLANDLewis CarrollTHE MILLENNIUM FULCRUM EDITION 3.0CHAPTER I. Down the Rabbit-HoleAlice was beg ^ 1); but when the Rabbit actually TOOK A WATCHOUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice start ; then shelooked at the sides of the well, and noticed that they were filled withcupboards and book-shelves; here and there the schoolroom, and though thiswas not a VERY good opportunity for showing off her knowledge, as therewas no one to listen aps I shall see it written up somewhere. 'Down, down, down. There was nothing else to do, so Alice soon begantalking again. Alice was not a bit hurt, and she jumped up on to her feet in a moment: she looked up, but it was all dark overhead; before , alas! either the locks were too large, or the key was too small, but at any rate it would not open any of them. However, o few things indeed were reallyimpossible. There seemed to be no use in waiting by the little door, so she wentback to the tab e had neverforgotten that, if you drink much from a bottle marked 'poison,' it isalmost certain to disagree with you, soone like then?' And she tried to fancy what the flame of acandle is like after the candle is blown out, for she could not remem gcheated herself in a game of croquet she was playing against herself, for this curious child was very fond of pretending to othing but out-of-the-waythings to happen, that it seemed quite dull and stupid for life to go onin the common way. So she s nd she went on planning to herself how she would manage it. 'They mustgo by the carrier,' she thought; 'and how funny it'll and reaching half down the hall. After a time she heard a little pattering of feet in the distance, and she hastily dried her I got up this morning? I almost think I can remember feeling alittle different. But if I'm not the same, the next question forMabel! I'll try and say "How doth the little -- " and she crossed herhands on her lap as if she were saying lessons, and ,if I like being that person, I'll come up: if not, I'll stay down heretill I'm somebody else"--but, oh dear!' cried Alice, ! the little door was shut again, and the little golden key waslying on the glass table as before, 'and things are worse th , I suppose, bybeing drowned in my own tears! That WILL be a queer thing, to be sure!However, everything is queer to-day.'J d to her to wink with one of its little eyes, but it said nothing. 'Perhaps it doesn't understand English,' thought Alice; 'I and she sits purring so nicely by the fire, licking her paws andwashing her face--and she is such a nice soft thing to nurs 't remember half of them--and it belongs to a farmer, you know, and he says it's so useful, it's worth a hundred pounds! Hes e shore.CHAPTER III. A Caucus-Race and a Long TaleThey were indeed a queer-looking party that assembled on the bank--thebir er eyes anxiously fixed on it, for she feltsure she would catch a bad cold if she did not get dry very soon. 'Ahem!' said th

OVERVIEW OF PROCESS:

We have learnt the dependencies between characters and the conditional probabilities of characters in sequences so that we can in turn generate wholly new and original sequences of characters.

We have used, Keras which is an open source neural network library written in Python. It is capable of running on top of MXNet, Deeplearning4j, Tensorflow, CNTK or Theano

```
CODE:
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.callbacks import ModelCheckpoint
from keras.utils import np utils
# load ascii text and covert to lowercase
filename = "wonderlandcopy.txt"
raw_text = open(filename).read()
raw_text = raw_text.lower()
# create mapping of unique chars to integers
chars = sorted(list(set(raw text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))
n_chars = len(raw_text)
n vocab = len(chars)
print ("Total Characters:", n_chars)
print ("Total Vocab: ",n_vocab)
# prepare the dataset of input to output pairs encoded as integers
seq length = 100
dataX = []
dataY = []
for i in range(0, n_chars - seq_length, 1):
seq in = raw text[i:i + seq length]
```

```
seq out = raw text[i + seq length]
dataX.append([char_to_int[char] for char in seq_in])
dataY.append(char_to_int[seq_out])
n patterns = len(dataX)
print ("Total Patterns: ", n patterns)
# reshape X to be [samples, time steps, features]
X = numpy.reshape(dataX, (n patterns, seq length, 1))
# normalize
X = X / float(n vocab)
# one hot encode the output variable
y = np utils.to categorical(dataY)
# define the LSTM model
model = Sequential()
model.add(LSTM(150, input shape=(X.shape[1], X.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(v.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam')
# define the checkpoint
filepath="weights-improvement-{epoch:02d}-{loss:.4f}.hdf5"
                    ModelCheckpoint(filepath, monitor='loss', verbose=1,
checkpoint
save best only=True, mode='min')
callbacks_list = [checkpoint]
model.fit(X, y, epochs=20, batch size=128, callbacks=callbacks list)
# load the network weights
filename = "weights-improvement-14-2.1243.hdf5"
```

```
model.load weights(filename)
model.compile(loss='categorical_crossentropy', optimizer='adam')
# pick a random seed
start = numpy.random.randint(0, len(dataX)-1)
pattern = dataX[start]
print ("Seed:")
print ("\"",".join([int_to_char[value] for value in pattern]), "\"")
# generate characters
for i in range(1000):
x = numpy.reshape(pattern, (1, len(pattern), 1))
x = x / float(n_vocab)
prediction = model.predict(x, verbose=0)
index = numpy.ndarray.argmax(prediction)
result = int_to_char[index]
print ("new result", result)
seq_in = [int_to_char[value] for value in pattern]
sys.stdout.write(result)
pattern.append(index)
pattern = pattern[1:len(pattern)]
print ("\nDone.")
```

OUTPUTS:

Error Obtained from the 15th Epoch: 2.1243

```
Using TensorFlow backend.
Total Characters: 11440
Total Vocab: 45
Total Patterns: 11340
Epoch 1/20
3, saving model to weights-improvement-00-3.1285.hdf5
Epoch 2/20
3235, saving model to weights-improvement-01-3.0324.hdf5
11340/11340 [============] - 63s - loss: 3.0324
1704, saving model to weights-improvement-02-3.0170.hdf5
Epoch 4/20
11264/11340 [=============================] - ETA: Os - loss: 3.0170Epoch 00003: loss improved from 3.01704 to 3.0
1612, saving model to weights-improvement-03-3.0161.hdf5
11340/11340 [============= ] - 68s - loss: 3.0161
Epoch 5/20
1150, saving model to weights-improvement-04-3.0115.hdf5
11340/11340 [===========] - 66s - loss: 3.0115
Epoch 6/20
11264/11340 [============================] - ETA: Os - loss: 3.0056Epoch 00005: loss improved from 3.01150 to 3.0
0563, saving model to weights-improvement-05-3.0056.hdf5
Epoch 7/20
0050, saving model to weights-improvement-06-3.0005.hdf5
```

```
■ Terminal File Edit View Search Terminal Help
                                                                                                                                                                     🌕 📶 🛅 🛱 (0:42, 82%) 🕪 Mon Oct 30 2017 11:40 🖳
            start = numpy.random.randint(0, len(dataX)-1)
            pattern = dataX[start]
             print ("Seed:'
print ("\"", '
                                   '.join([int_to_char[value] for value in pattern]),
             ^Ix = numpy.reshape(pattern, (1, len(pattern), 1))
            ^Ix = x / float(n_vocab)
^Iprediction = model.predict(x, verbose=0)
            'Alindex = numpy.argmax(prediction)

'Alresult = int_to_char[index]

'Alseq_in = [int_to_char[value] for value in pattern]

'Alsys.stdout.write(result)

'Alpattern.append(index)

'Alpattern.append(index)
             ^Ipattern = pattern[1:len(pattern)]
Seed:
 are your shoes done with?' said the gryphon. 'i mean, what
alice looked down "
an the pabbit sase the waite oo the garte oa the botr and the cadl. and the was aolng the woile she whit sae in the coulo,
'hh yhu a latg toine i sas a little 'iite ' said the daterpillar.
'ie cou dot yhu,' said the kacte haree
tai ionttenning to the tabbit and whit she was to torl to the tooe and whst hlr lasde whtt the rooer oh the tooee
'hh you dan to toink i sas a datter an i saa -tte toine i sas t yhul toe 'orld 'hu, and the mort oaad to the toiee. and the wait to the woile ald whs
t hlr looten the woole
'hh you dan to tan i toile 'hu, a dan and go whe aad the wai int in the garter, and the was aolng the woile she whit she was to the whrt she was oo th
e tooee, and she taid the was oo the tooe, and she taid the was oo the tooe, and she taid the was oo the gorr, and she tai iot io whet the was to the
 woole
'ho wou dan to tay 'hu, and then io the darter
she harter was toe tiet tored an io was in the carter, and she was aolng the woile she whit she was to the whrt she white wa
Done.
```

REFERENCES:

- 1. SEPTEMBER 17, 2015 BY DENNY BRITZ
 Recurrent Neural Networks Tutorial, Part 1 Introduction to RNNs
 http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- 2. This blog is important : Understanding LSTM Networks : http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- 3. The Unreasonable Effectiveness of Recurrent Neural Networks http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- 4. Videos : Recurrent Neural Networks (RNN / LSTM)with Keras Python https://www.youtube.com/watch?v=4rG8lsKdC3U
- 5. KERAS Documentation

https://keras.io/