Cheryl Ngo
12/1/2019

1. Pretreating the data
   The model is trained using a data set of news headlines. First, the data is loaded and 10,000 headlines are chosen at random for use in training the model. We set the parameters in the following section including epochs, input sequence length, dense layer quantity, and hidden unit quantity. Next, the data is combined into one string so it can be tokenized and formatted into groups of seq_length units and a categorical response variable is stored.

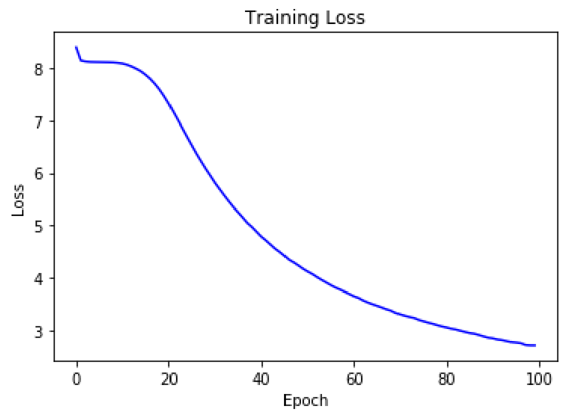
   The model
   The RNN is created and trained saving checkpoints every 10 epochs and the loss of the model through the training process is graphed. This model consists of an embedding layer with a 10-dimensional projection, an LSTM layer with 100 hidden units, and dense layer with softmax activation. LSTM was used as to mitigate the vanishing gradient problem and softmat was chosen as the activation since the output should be a probability due to our categorical outcome. Finally, we can choose a checkpoint to load weights from. We generate an output by using predict_classes to feed a sequence of words into our model and generate the model's prediction of the output word.

2. Implement an RNN
   After several test runs using a small subset of the training data (around 200 samples), the first model I tried used 10,000 samples from the headlines with a batch size of 128 trained for 50 epochs. It took 25 minutes and 20 seconds to train and produced headlines at each checkpoint that were mostly the word "to." Then, I trained a model with a batch size of 50 for 50 epochs. The smaller batch size resulted in headlines with mostly unique words after only 30 epochs. To confirm this, I tried the same model with a batch size of 20 through 40 epochs; this took 46 minutes and 42 seconds and none of the resulting headlines contained only unique words. I found that many of my models produced a repetitious string of the word "to" and that training it for more epochs and changing the batch size helped the model expand beyond this repetition. I expected that the model with a batch size of only 20 would have had more unique words in its 30-epoch checkpoint that the model with 50 in each batch, but this change did not seem to happen. As a result of all this testing, I ran another intensive model of the 10,000 samples with a batch size of 128 trained for 100 epochs. This took 23 minutes and 18 seconds to train and resulted in the outputs below:

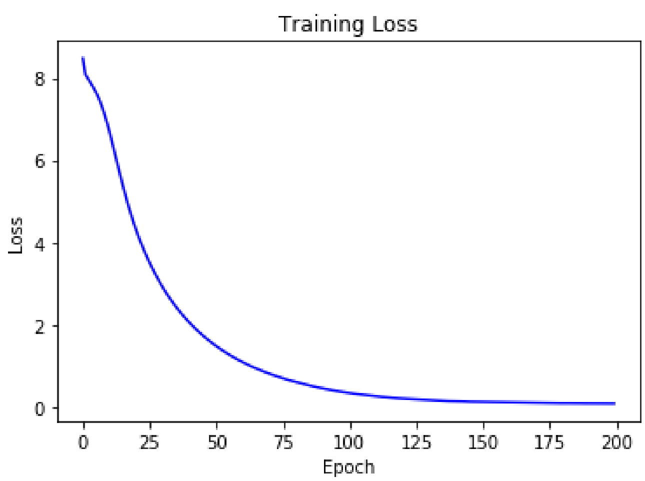
Epoch	Seed	Output
20	adds to milans san siro blues spain arrests 14 linked	to to to to
40	re form lobby group still hopes for hospital site rethink	migratory shorebird leaching for
60	may give rookie a chance an asia pacific interview with	bn on think powerline sixers replica to

Cheryl Ngo
12/1/2019

80	plant consultation albany bunbury gas pipeline way overdue says labor	taxi driver to bury spray is
100	nigel scullion apologises for abuse at don dale detention centre	rebels mcmahon open to

a. 

3. From the results above, I found that smaller batch made for smaller loss and smaller loss made for more words being used besides "to." Then, I added embedding to the model. Embedding resulted in a more natural use of language, but it was hard to judge the results generated by the model in terms of natural language flow. I trained the model with the embedding layer through 200 epochs. It took 1 hour and 9 minutes to train. I selected checkpoints within the first 100 epochs to display below:

4. 

The training loss for this model starts improving minimally after around 100 epochs.

5. I used checkpoints within the 100 epochs specified above to generate outputs from the model

Epoch	Seed	Output
20	new bushfire seafety campaign	png matthewson binge fisk reports

40	back scheme still alive to fight	hwy burnside girl divorce
60	limit cannes line up	mystery keeps world cup winners tasmanians snap
80	great koala national park	aerosmith cancels troubled n america tour court
100	need grounding brown syria	proposes wmd free zone bad virgin blue

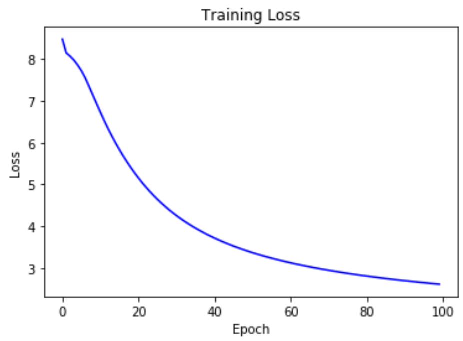
In searching for the predictive text results in the original data file, it seems that the checkpoints from higher epochs put together words into phrases that appear in the full data. For example, in epochs 50, “massive elephant bird egg” appears in two headlines so it would make sense for the model to learn that these words are associated. Similarly, in epoch 100, “free zone” and “virgin blue” are commonly associated words in the data.

6. Headlines to not use typical English grammar and tent to omit words and create abbreviations in favor of brevity. Similarly, different headlines could be present in one sequence since each headline does not have a set length, so I would not expect this model to produce grammatically correct sentences. I decided to use my original model with double the amount of hidden units in the LSTM layer trained from 25,000 samples for 110 epochs. It took 2 hours and 4 minutes to train and generated the 15 outputs below:

Seed	Output
african pride through fashion	unis in taiwan british was killed
leader detained in raid	witnesses us shoalhaven oil to end
23 killed in indian	ambush alp sues longest lucky art to help
afl interviews jimmy bartel	brad ottens steven noodle court
hopes tight budget will	slash producers not expected to
as the hawks face	the new medical salmonella plant
for cross border bushfire	irrigators to extend off soon
armstrong medal back in	ioc possession for latest trade
beveridge asking hard questions	of best bail on burke freeze fees
victorian same sex couples	show federation to weekend funding on june
netball lawn bowls shine	in asc costs eu

Cheryl Ngo
12/1/2019

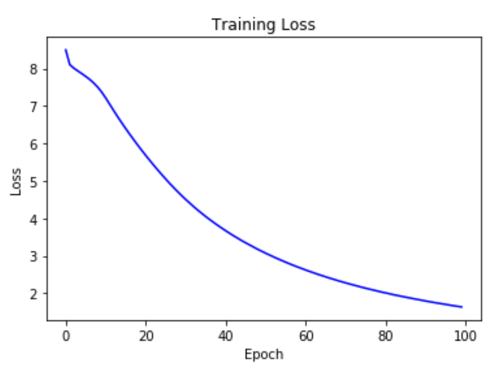
the balance rain washes	out on northern spirits
damaging an phils attacks	hiding not worried to

i. 

7. Testing model variations
    a. Double hidden units: In this model, I changed the hidden units in the LSTM layer from 200 to 100 and trained it though 100 epochs. It took 44 minutes and 15 seconds to train and produced the following output:

Epoch	Seed	Output
20	south coast drivers steer	in crash in crash police
40	rebuilding scheme hope for	unesco fast abuse postal ny trial in assess
60	up double misery for	candidacy rates accusations to lift
80	meeting fails to back	rex minerals new qld times
100	decision former saddam bodyguards	case re examined allsopp still committed

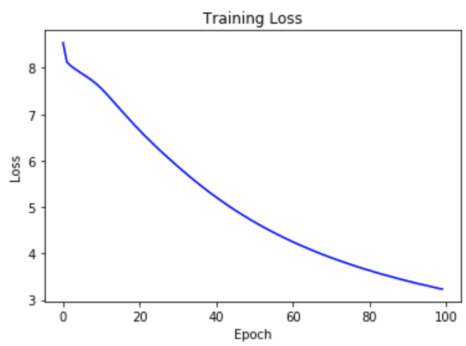
I think that the output from this model shows an improvement over the first model, maintaining a low loss after 100 epochs without overtraining. It seems like these are grammatically better (even more so if punctuation is added) than the first model.

i. 

b. Halve hidden units: In this model, I changed the hidden units in the LSTM layer from 200 to 400 and trained it though 100 epochs. It took 35 minutes and 38 seconds to train and produced the following results

Epochs	Seed	Output
20	lends 200m to goodman	a assault from the league port
40	to face home of	national meeting star wars
60	rangers hit five celtic	zealander for liability advertising
80	proud brazilian record premier	praises in north crash malaysian the aurizon aimed
100	of bishop after super	blunder king in the ring by chess

This model could have probably trained for longer to produce improved results, but the other models showed more promising results in terms of the readability of their output after a similar loss statistic and fewer epochs.

i. 

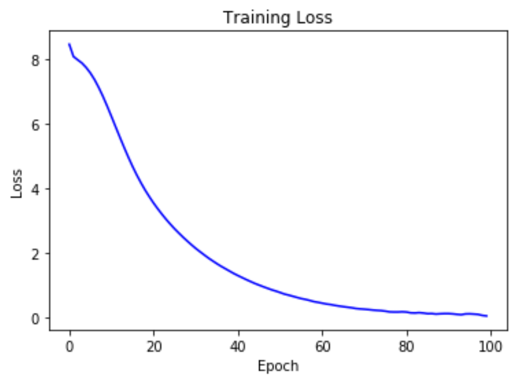
c. Double sequence length: In this model, I changed the sequence length from 4 to 8 and trained it through 100 epochs. It took 1 hour and 3 minutes to train and produced the results below

Epochs	Seed	Output
20	kills 3 nsw government urged to make drivers	licenses more attainable f household
40	contracts put leaks in water plan fishermen found	safe to be lpg rips driver backs another
60	for un greece votes driver doing burnouts before	fatal crash unis could face
80	nine dead after fighting in gaza esso fined over	longford gas leak brothers face attempted murder

Cheryl Ngo
12/1/2019

100	sparrow its time we had the sex talk with	the literary review cassidy

This model seemed to perform decently in terms of the readability of the output. After about 70 epochs it appears to begin overtraining, so I examined mostly the results of the checkpoint from 60 and 80 epochs. I don't think that the longer sequence length is appropriate for this model due to the short nature of headlines which resulted in outputs that made some sense for the first few words and then deviated.

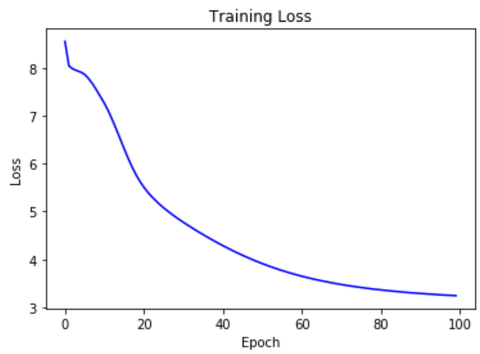
i. 

d. Halve sequence length: In this model, I changed the sequence length from 4 to 2 and trained it through 100 epochs. It took 47 minutes and 12 seconds to train and produced the results below:

Epochs	Seed	Output
20	chain boost	to be consulted over fatal crash on the
40	bill childcare	centre of the drum wednesday 28th november
60	israeli assault	of the drum wednesday
80	leak on	the drum wednesday february kathy jackson doctor predicts
100	joins cfmeu	to be made unmissables

Despite training for 100 epochs and very different seeds, there were a few common words through several of the outputs which leads me to believe that the shorter sequence length results in homogenized outputs due to the lack of fuller context for the model.

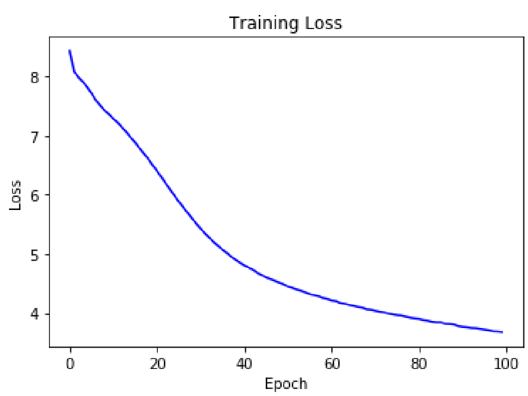
Cheryl Ngo
12/1/2019



i.

e. Add a dense layer: In this model, I added a relu activated dense layer with 50 units with dropout before the original dense layer. It took 48 minutes and 28 seconds to train and produced the output below

Epochs	Seed	Output
20	treatment annan wants tsunami	cup in the analysis in the the
40	parts of princes hwy cancer	writing bench group to take in
60	reverse same sex premier	concedes consultant paid 1 evans of medicare
80	over bieber crowd crush	cold case govt birney to explain share
100	roads leave 8m bill	tasmanians volunteer pilot a



i.

8. The best headline was "case re examined allsopp still committed" from the seed "decision former saddam bodyguards". The funniest headline was "to be made unmissables ."

10. Overall, I found that using embeddings made a big difference in my model's ability to produce outputs that made some sense early in the training process. If I were to continue this research, I would want to determine how I could use predictive text for headlines without combining different headlines into a single input element—possibly by truncating headlines or using padding differently—so that the model does not learn the combination of the end word of one line and the start word of another line as a possibly significant pairing.

Cheryl Ngo
12/1/2019

Appendix

```
"""
Cheryl Ngo

12/1/19
"""
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Code Flow
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
'''
The model is trained using a data set of news headlines. First, the data
is loaded and 10,000 headlines are chosen at random for use in training
the model.  We set the parameters in the following section including
epochs, input sequence length, dense layer quantity, and hidden unit
quantity. Next, the data is tokenized and formatted into groups of
seq_length units and a categorical response variable is stored.  The
RNN is created and trained saving checkpoints every 10 epochs and the loss
of the model through the training process is graphed.  Finally, we can
choose a checkpoint to load weights from, generate a starting seed, and
output our models prediction for the next word.
'''
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Import Libraries Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
from numpy import array
from keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding

import random as rand
import numpy as np
import pandas as pd
from keras.callbacks import ModelCheckpoint
import datetime
import matplotlib.pyplot as plt
from keras.layers import Dropout
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Variables Used
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
'''
text is used to store the subset of headlines used to train the model
raw_text formats this into one string
```

start_time and end_time are used to calculate training time for the model

doubleHidden doubles the hidden units in the LSTM layer
halveHidden halves the hidden units in the LSTM layer

doubleSeq doubles the sequence length of the random seed used to generate text
halveSeq havles the sequence length of the random seed used to generate text

addDense adds another dense layer to the model

num_epoch is used to specify the number of epochs for which the model will be trained
batch_sz is used to specify the batch size used when fitting the model

tokenizer is the raw_text without punctuation and tokenized
int_to_word is used to convert the integers feed to the model back to words

vocab_size is the amount of unique words in text
sequences is used to store the data in groups of length seq_length used to predict the next word
X contains the groups used to predict the next work
y contains the actual next words

path is used to specify where the checkpoints will be saved
filepath specifies the name of the checkpoint file

checkpoint and callbacks_list allow the checkpoints to be accesssed later

pattern is the specified group of words used seed the model
seed_text converts pattern from integers to words
   seed_text is also updated at the end of the code to include the word predictions

encoded is used to store seed_text converted back into integers and padded
yhat stores the predition of the next word given the encoded sequence

```
'''
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Objective
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
'''The goal of this program is to create a recurrent neural network that will
generage headlines of four to eight words for articles
'''
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Load Data Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
#Import the data
#Fix random weights for repeatability
np.random.seed(7)

#Import the data
```

Cheryl Ngo
12/1/2019

```
dataframe=pd.read_csv('C:\\Users\\chery\\Grad School\\BUAD 5802 -AI II\\M5-Deep Learning for
Text and Sequences\\Submission\\abcnews-date-text.csv')
#Keep only the headline column
text=list(dataframe['headline_text'])

#Due to memory error and speed selecting only 10,000 headlines
text=rand.sample(text,25000)

#Capture the time to allow us to calculate training duration
start_time = datetime.datetime.now()
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Parameters Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
#Use the following true/false variables to vary the model
#Modify the hidden units in the LSTM Layer
doubleHidden=True
halveHidden=False

#Modify the sequence seed length
doubleSeq=False
halveSeq=False

#Modify the number of dense layers
addDense=False

#The following define the modified units
#Set the hidden units in the LSTM layer
if doubleHidden:
   hidden_units=100
elif halveHidden:
   hidden_units=50
else:
   hidden_units=200

#Set the sequence length for seeding
if doubleSeq:
   seq_length=8
elif halveSeq:
   seq_length=2
else:
   seq_length = 4

#Set the number of epochs for training
num_epoch=100
#Set the batch size
batch_sz=128

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
```

Cheryl Ngo
12/1/2019

```
Pretreat Data Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
#Format the imported text as one string
raw_text=''
for i in text:
   #Combine each headline with a space in between
   raw_text+=' '
   #Make all the text lowercase
   raw_text+=i.lower()

#Tokenize
tokenizer=Tokenizer()
tokenizer.fit_on_texts([raw_text])
encoded=tokenizer.texts_to_sequences([raw_text])[0]

#Create a dictionary to convert the values back to words
int_to_word = dict((c, i) for i, c in tokenizer.word_index.items())

# determine the vocabulary size
vocab_size = len(tokenizer.word_index) + 1
print('Vocabulary Size: %d' % vocab_size)

# encode seq_length words -> 1 word
sequences = list()
for i in range(seq_length-1, len(encoded)):
        sequence = encoded[i-seq_length+1:i+1]
        sequences.append(sequence)
print('Total Sequences: %d' % len(sequences))

# pad sequences
max_length = max([len(seq) for seq in sequences])
sequences = pad_sequences(sequences, maxlen=max_length, padding='pre')
print('Max Sequence Length: %d' % max_length)

# split into input and output elements
sequences = array(sequences)
X, y = sequences[:,:-1],sequences[:,-1]
y = to_categorical(y, num_classes=vocab_size)

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Define Model Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
#Define the LSTM model
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=max_length-1))
model.add(LSTM(hidden_units))

if addDense:
```

```
        model.add(Dense(50, activation='relu'))
        model.add(Dropout(0.25))
        model.add(Dense(vocab_size, activation='softmax'))

    else:
        model.add(Dense(vocab_size, activation='softmax'))

    model.summary()

    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    Train Model Section
    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    #Define the filepath in which to save the checkpoints
    path='C:\\Users\\chery\\Grad School\\BUAD 5802 -AI II\\M5-Deep Learning for Text and
    Sequences\\Submission\\'
    #Define the naming schema for the checkpoints
    filepath=path+"weights-improvement-{epoch:02d}-{loss:.4f}-dropout.hdf5"

    #Save the checkpoints
    checkpoint = ModelCheckpoint(filepath, save_weights_only=True, verbose=1, period=20)
    callbacks_list = [checkpoint]

    # compile network
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # fit network
    history=model.fit(X, y, epochs=num_epoch, batch_size=batch_sz, verbose=1, callbacks=callbacks_list)

    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    Show Output Section
    """"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
    #Save the stop time to calculate the run time
    stop_time = datetime.datetime.now()
    #Print the run time
    print ("Time required for training:",stop_time - start_time)

    #Graph of loss
    #Save the test loss of the model per epoch to plot
    loss = history.history['loss']
    #Save the number of epochs for use in our metric plot
    epochs = range(len(loss))

    #Create figure
    plt.figure()
    #Create a plot of training loss
    plt.plot(epochs, loss, 'b', label='Training loss')
    #Set a title for the plot
    plt.title('Training Loss')
    #Set a y-axis label for the plot
```

```
plt.ylabel('Loss')
#Set an x-axis label for the plot
plt.xlabel('Epoch')

#Show the plots
plt.show()

""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
Generate New Output Section
""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
#Specify which checkpoint to use for the output
filename = "weights-improvement-100-2.6093-dropout.hdf5"
filename=path+filename
#Load the specified checkpoint's weights
model.load_weights(filename)
#Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam')

#Select a random place to take the seed
start = np.random.randint(0, len(sequences)-1)
pattern = list(sequences[start])
#Print the seed sequence
seed_text=' '.join([int_to_word[value] for value in pattern])
print("Seed:")
print("\"", seed_text, "\"")

# generate a sequence from a language model
# generate a fixed number of words
for i in range(0,np.random.randint(4,9)):
	# encode the text as integer
	encoded = tokenizer.texts_to_sequences([seed_text])[0]
	# pre-pad sequences to a fixed length
	encoded = pad_sequences([encoded], maxlen=max_length-1, padding='pre')
	# predict probabilities for each word
	yhat = model.predict_classes(encoded, verbose=0)
	# map predicted word index to word
	out_word = ''
	for word, index in tokenizer.word_index.items():
		if index == yhat:
			out_word = word
			break
	# append to input
	seed_text += ' ' + out_word
print(seed_text)
```