REPORT

Assignment 4:

Building a character-level language model using RNN

(or LSTM or Bidirectional LSTM or GRU)

MR.CHALERMKIAT CHANAHCHAN

ID: 62070701602

King Mongkut's University of Technology Thonburi

Faculty of Engineering, Department of Computer Engineering

CPE 663 Special Topic: Deep Learning, 1/2019

Assignment 4: Building a character-level language model using RNN (or LSTM or Bidirectional LSTM or GRU).

Due: 21 November 2019. Please submit your report in PDF to LEB2

Introduction

In this assignment, you can build a character-level language model for any data source that you have chosen (e.g. Wikipedia, GitHub Code, Shakespear, Textbook, etc.). You can use any technique that design of the RNN as you please.

Tasks

1. Choose the data source for building a language model.
2. Design RNN (or derivatives) to model a character-level language model.
3. Train the model.
4. Evaluate the results.
5. Show the examples of text generated by the model.

Tasks 1: Choose the data source for building a language model.

I used the data source from https://www.patentsview.org/query/ and search with “bicycle” word.

Code for preparation data:

```
### load data bicycle csv to dataframe
    abstracts = pd.read_csv('/content/drive/My Drive/Colab Notebooks/bicycle_text
.csv')
    print ("Shape of Data sets: {0}".format(abstracts.shape))
    print ("Example of Data sets: \n{0}".format(abstracts['patent_abstract'][1]))
    print ("Amount of Null in patent_abstract row: {0}".format(abstracts['patent_
abstract'].isnull().sum()))
    Not_null_abstracts = abstracts.mask(abstracts.eq('None')).dropna()
    print ("After Removing Null in dataframe: \n{0}".format(Not_null_abstracts))
    clear_data = Not_null_abstracts
```

Shape of Data sets: (13467, 2)

Example of Data set:

The invention relates to a bracket for holding U shaped locks onto the seat posts of bicycles in a space efficient manner. One end of the bracket uses a circular shaped post attaching means which can be tightened around differently sized seat by means of a screw. The other end has a channel with a C-shaped cross section which is deformable and can be made to fit around a portion of the U lock. The shape of the channel is at right angles to shape of the seat post attaching means and thus the U lock can be neatly positioned under the seat of a bicycle.

Inside the raw data, I found many “None” and “NaN” inside the data, I remove it with **function dropna():**

	patent_abstract	patent_title
0	None	‘S’-shaped bicycle tube
1	The invention relates to a bracket for holding...	"""U"" post bracket for bicycles"
2	A handlebar brake for a motorcycle, motor bicy...	"Rear ""stop"" light contactor for a vehicle s...
3	A bicyclist helmet has a outer shell; an inner...	(Airlock) bicycle helment with adjustable vent...
4	The 10-speed bicycle has a single manually mov...	10-Speed bicycles
...	...	...
13462	Apparatus adapted for adjusting the position o...	Workpiece and handlebar adjustment apparatus
13463	A bicycle workstand is mounted to a support st...	Workstand for bicycles
13464	None	Wrench for a bicycle axle nut
13465	NaN	Yoke for an exercise bicycle
13466	A method of production of an α+β-type titanium...	α+beta-type titanium alloy part and method of ...

[13467 rows x 2 columns]

After Removing Null in Data Frame:

	patent_abstract	patent_title
1	The invention relates to a bracket for holding...	"""U"" post bracket for bicycles"
2	A handlebar brake for a motorcycle, motor bicy...	"Rear ""stop"" light contactor for a vehicle s...
3	A bicyclist helmet has a outer shell; an inner...	(Airlock) bicycle helment with adjustable vent...
4	The 10-speed bicycle has a single manually mov...	10-Speed bicycles
5	The present invention provides, in its princip...	1H-4(5)-substituted imidazole derivatives
...	...	...
13460	On an upright support is an apparatus includin...	Work stand for bicycles
13461	A workout apparatus for use by a user to simul...	Workout apparatus for simulating user movement...
13462	Apparatus adapted for adjusting the position o...	Workpiece and handlebar adjustment apparatus
13463	A bicycle workstand is mounted to a support st...	Workstand for bicycles
13466	A method of production of an α+β-type titanium...	α+beta-type titanium alloy part and method of ...

[11203 rows x 2 columns]

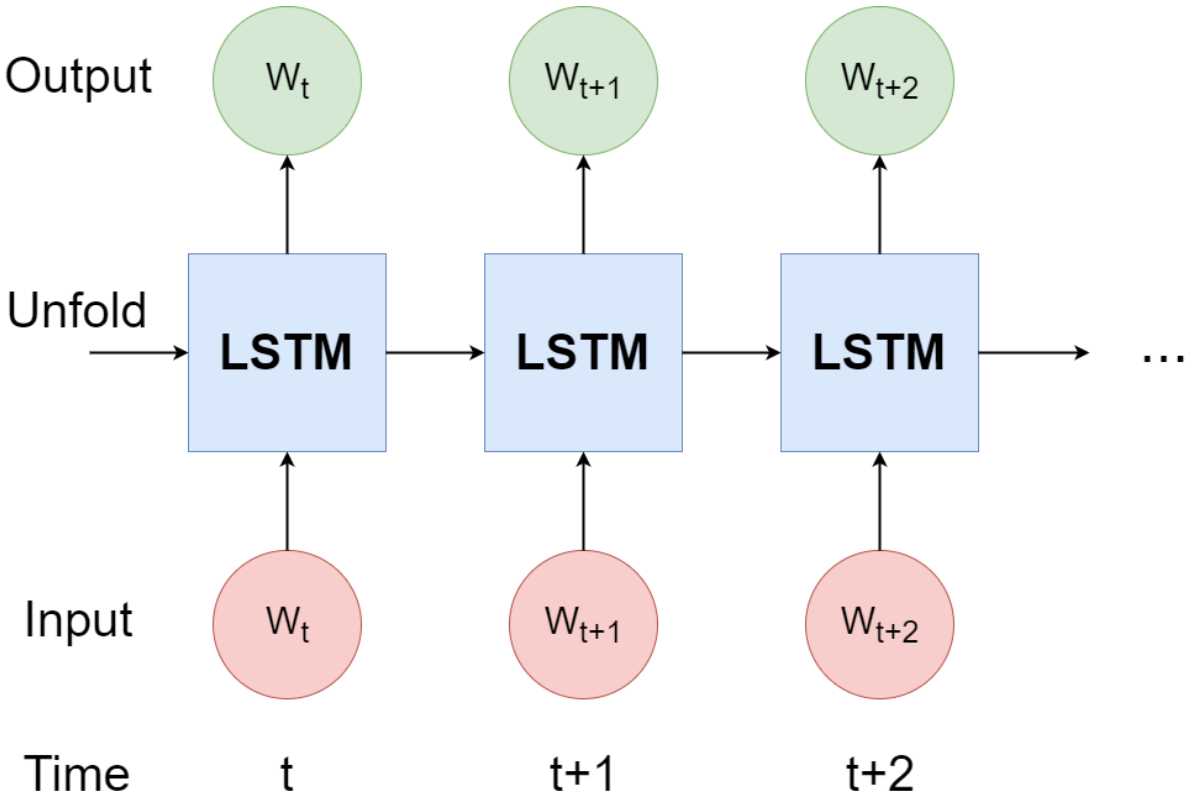
And Select only "Patent_abstract" column for using training model.

1	The invention relates to a bracket for holding...
2	A handlebar brake for a motorcycle, motor bicy...
3	A bicyclist helmet has a outer shell; an inner...
4	The 10-speed bicycle has a single manually mov...
5	The present invention provides, in its princip...
...	...
13460	On an upright support is an apparatus includin...
13461	A workout apparatus for use by a user to simul...
13462	Apparatus adapted for adjusting the position o...
13463	A bicycle workstand is mounted to a support st...
13466	A method of production of an α+β-type titanium...

Name: patent_abstract, Length: 11203, dtype: object

Note: In this assignment 4, I use only 300 data from 1 to 300 because I already tried to use all of data sets, and it used a lot of times 8 hour each epochs).

Tasks 2: Design RNN (or derivatives) to model a character-level language model.



Tasks 3: Train the model.

Code for creating the model:

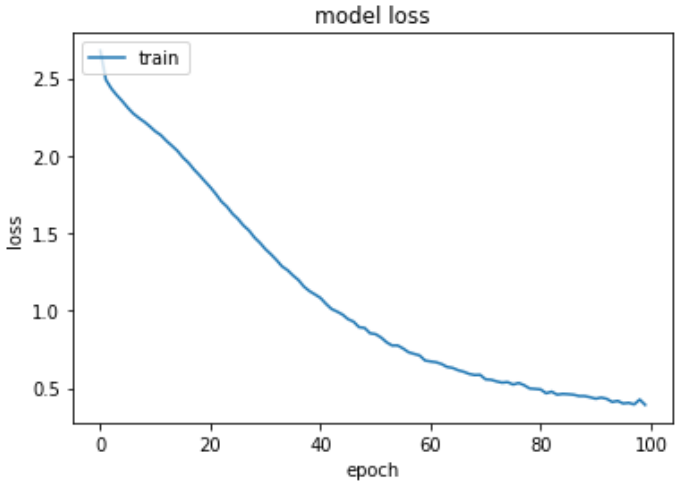
```
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout

def create_model(predictors, label, max_sequence_len, total_words):

    model = Sequential()
    model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
    model.add(LSTM(150, return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(100))
    model.add(Dense(total_words, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    history = model.fit(predictors, label, epochs=100, verbose=1)
    print (model.summary())
    return model, history
```

Model Summary:

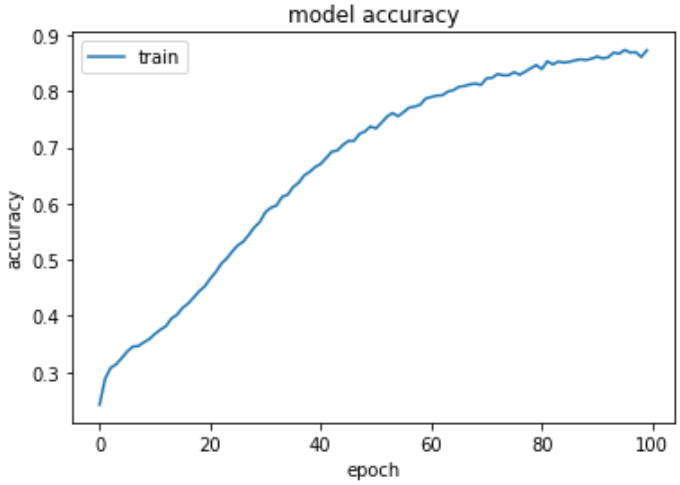
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 129, 100)          6000
_________________________________________________________________
lstm_1 (LSTM)                (None, 129, 150)          150600
_________________________________________________________________
dropout_1 (Dropout)          (None, 129, 150)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               100400
_________________________________________________________________
dense_1 (Dense)              (None, 60)                6060
=================================================================
Total params: 263,060
Trainable params: 263,060
Non-trainable params: 0
_________________________________________________________________
```

Tasks 4: Evaluate the results.

Graph: Loss value each epochs



Graph: Accuracy value each epochs



Tasks 5: Show the examples of text generated by the model.

Code:

```
def generate_text(seed_text, next_words, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted = model.predict_classes(token_list, verbose=0)
        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text
```

Example of text generated:

```
generate_text("A bicycle for the", 100, max_sequence_len)
```

Output:

'A bicycle for the of the bicycle a in a and at one end to the seat of the bicycle a to and the and a of to the and to the of the bicycle to a of for the to the bicycle to of the of the of the in the of the to bicycle is made of the to onto the of the in the of the to thus is to a for a bicycle is the can onto the in of the invention has and by the the is by the and the to be of the to bicycle for with'

```
generate_text("The invention relates to a bracket for holding", 100, max_sequence_len)
```

Output:

'The invention relates to a bracket for holding u shaped locks onto the seat posts of bicycles in a space efficient manner one end of the bracket uses a circular shaped post attaching means which can be tightened around differently sized seat by means of a screw the other end has a channel with a c shaped cross section which is deformable and can be made to fit around a portion of the u lock the shape of the channel is at right angles to shape of the seat post attaching means and thus the u lock can be neatly positioned under the seat of a bicycle a'

Full code in this assignment 4:

```
# -*- coding: utf-8 -*-
"""Assigment4.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1kFuqFBJbv0fL3y9f-DQAgNncjd2lVOuV
"""
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
import keras.utils as ku
import numpy as np
import pandas as pd

tokenizer = Tokenizer()

def dataset_preparation():
    # basic cleanup
    ### load data bicycle csv to dataframe
    abstracts = pd.read_csv('/content/drive/My Drive/Colab Notebooks/bicycle_text.csv')

    print ("Shape of Data sets: {0}".format(abstracts.shape))
    print ("Example of Data sets: \n{0}".format(abstracts['patent_abstract'][1]))
    print ("Amount of Null in patent_abstract row: {0}".format(abstracts['patent_abstract'].isnull().sum()
))
    Not_null_abstracts = abstracts.mask(abstracts.eq('None')).dropna()
    print ("After Removing Null in dataframe: \n{0}".format(Not_null_abstracts))

    clear_data = Not_null_abstracts

    # tokenization
    tokenizer.fit_on_texts([clear_data['patent_abstract'][1]])
    total_words = len(tokenizer.word_index) + 1
    print(total_words)
    print ("Clear_data for sequences: \n{0}".format(clear_data['patent_abstract']))
    input_sequences = []
    for sentence in clear_data['patent_abstract'][:300]:
        token_list = tokenizer.texts_to_sequences([sentence])[0]

        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i+1]
            input_sequences.append(n_gram_sequence)

    #Find Maximum of sequences
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
    # create predictors and label
    print (input_sequences)
    predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
    label = ku.to_categorical(label, num_classes=total_words)

    return predictors, label, max_sequence_len, total_words

from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout

def create_model(predictors, label, max_sequence_len, total_words):

    model = Sequential()
    model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
    model.add(LSTM(150, return_sequences = True))
    model.add(Dropout(0.2))
    model.add(LSTM(100))
    model.add(Dense(total_words, activation='softmax'))

    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
    history = model.fit(predictors, label, epochs=100, verbose=1)
    print (model.summary())
    return model, history

def generate_text(seed_text, next_words, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted = model.predict_classes(token_list, verbose=0)

        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text

predictors, label, max_sequence_len, total_words = dataset_preparation()

model, history = create_model(predictors, label, max_sequence_len, total_words)

generate_text("A bicycle", 50, max_sequence_len)

import matplotlib.pyplot as plt
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['acc'])
# plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('/content/drive/My Drive/Colab Notebooks/Acc-1.png')

# 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()
plt.savefig('/content/drive/My Drive/Colab Notebooks/Loss-1.png')

generate_text("A bicycle for the", 100, max_sequence_len)

generate_text("The invention relates to a bracket for holding", 100, max_sequence_len)
```

Reference:

- https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470
- https://medium.com/@shivambansal36/language-modelling-text-generation-using-lstms-deep-learning-for-nlp-ed36b224b275
- https://keras.io/preprocessing/text/#tokenizer
- https://www.patentsview.org/query/