Assignment10SutowBrett

July 21, 2021

[2]:
```
#Assignment 10.1A#

import string

def tokenize(sentence):
    # Split the sentence by spaces
    words = sentence.split()
    table = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(table) for w in words]
    return stripped

sentence = "Good Morning, I hope you are having a great day Professor! Please␣
 ↪give me an A+"
tokens = tokenize(sentence)
print(type(tokens))
print(tokens)
```

```
<class 'list'>
['Good', 'Morning', 'I', 'hope', 'you', 'are', 'having', 'a', 'great', 'day',
'Professor', 'Please', 'give', 'me', 'an', 'A']
```

[3]:
```
#Assignmet 10.1B#
import string
import nltk

def ngram(tokens, n):
    # Split the sentence by spaces
    sentence = tokens.split()
    # Remove punctuation
    newtable = str.maketrans('', '', string.punctuation)
    stripped = [w.translate(newtable) for w in sentence]
    ngrams = nltk.ngrams(stripped, n)
    return ngrams

paragraph = "Fine Professor, I will take an A- instead! But, that is the lowest␣
 ↪I am going!"
ngrams = ngram(sentence, 1)
```

```
for g in ngrams:
    print(g)
```

```
('Good',)
('Morning',)
('I',)
('hope',)
('you',)
('are',)
('having',)
('a',)
('great',)
('day',)
('Professor',)
('Please',)
('give',)
('me',)
('an',)
('A',)
```

[4]:
```
#Assignment 10.1C#
import string
import nltk
from numpy import array
from numpy import argmax
from keras.utils import to_categorical


def one_hot_encode(tokens):
    results = array(tokens)
    print(results)
    results = to_categorical(data)
    return results

data = [1,2,4]
value = one_hot_encode(tokens)
print(value)
```

```
['Good' 'Morning' 'I' 'hope' 'you' 'are' 'having' 'a' 'great' 'day'
 'Professor' 'Please' 'give' 'me' 'an' 'A']
[[0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1.]]
```

[15]:
```
#Assignment 10.2#
#Not sure why I am getting an error, as I have this file#
#I have tried various ways to pull the file each time with this error appearing#
```

```
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
import os

from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense

imdb_dir = Path('dsc650/data/external/imdb/aclImdb')
training_samples = 100
maxlen = 50
max_words = 500
embedding_dim = 50
max_features = 1000



training_samples = 50
validation_samples = 100


model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
test_dir = os.path.join(imdb_dir, 'test')
labels = []
texts = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(test_dir, label_type)
    for fname in sorted(os.listdir(dir_name)):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname))
            texts.append(f.read())
            f.close()
            if label_type == 'neg':
                labels.append(0)
            else:
                labels.append(1)
x_train = data[:training_samples]
y_train = labels[:training_samples]
x_val = data[training_samples: training_samples + validation_samples]
y_val = labels[training_samples: training_samples + validation_samples]
```

```
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_data=(x_val, y_val))
```

```
sequences = tokenizer.texts_to_sequences(texts)
x_test = pad_sequences(sequences, maxlen=maxlen)
y_test = np.asarray(labels)
```

```
model.load_weights('pre_trained_glove_model.h5')
model.evaluate(x_test, y_test)
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_9 (Embedding)      (None, 50, 50)            25000
_________________________________________________________________
flatten_1 (Flatten)          (None, 2500)              0
_________________________________________________________________
dense_10 (Dense)             (None, 32)                80032
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 33
=================================================================
Total params: 105,065
Trainable params: 105,065
Non-trainable params: 0
_________________________________________________________________
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-15-fe70599bca4d> in <module>
     34 for label_type in ['neg', 'pos']:
     35     dir_name = os.path.join(test_dir, label_type)
---> 36     for fname in sorted(os.listdir(dir_name)):
     37         if fname[-4:] == '.txt':
     38             f = open(os.path.join(dir_name, fname))
```

```
FileNotFoundError: [Errno 2] No such file or directory: 'dsc650/data/external/
 ↪imdb/aclImdb/test/neg'
```

[16]:
```
#Assignment 10.3#
#It for some reason kept pulling up this error, I could not figure out how to␣
 ↪solve it. From what I saw there is nothing online#
#I followed the code directly how it is written in the book#


from keras.layers import LSTM



input_train = data[:training_samples]
y_train = labels[:training_samples]
input_test = data[training_samples: training_samples + validation_samples]
y_test = labels[training_samples: training_samples + validation_samples]


model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-2a668f690d02> in <module>
     21               loss='binary_crossentropy',
     22               metrics=['acc'])
---> 23 history = model.fit(input_train, y_train,
     24                     epochs=10,
     25                     batch_size=128,

/opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/training.
 ↪py in fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
 ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
 ↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
 ↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1038       # `Tensor` and `NumPy` input.
```

```
   1039       (x, y, sample_weight), validation_data = (
-> 1040           data_adapter.train_validation_split(
   1041               (x, y, sample_weight), validation_split=validation_split)
   1042

/opt/conda/lib/python3.8/site-packages/tensorflow/python/keras/engine/
 ↪data_adapter.py in train_validation_split(arrays, validation_split)
   1355   unsplitable = [type(t) for t in flat_arrays if not _can_split(t)]
   1356   if unsplitable:
-> 1357     raise ValueError(
   1358         "`validation_split` is only supported for Tensors or NumPy "
   1359         "arrays, found following types in the input: {}".
 ↪format(unsplitable))

ValueError: `validation_split` is only supported for Tensors or NumPy arrays,␣
 ↪found following types in the input: [<class 'int'>, <class 'int'>, <class␣
 ↪'int'>]
```

[18]:
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

model = Sequential()
model.add(layers.Embedding(max_features, 128, input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(lr=1e-4),
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_12 (Embedding)     (None, 500, 128)          1280000
_________________________________________________________________
```

```
conv1d_6 (Conv1D)            (None, 494, 32)           28704
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 98, 32)            0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 92, 32)            7200
_________________________________________________________________
global_max_pooling1d_3 (Glob (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 33
=================================================================
Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
157/157 [==============================] - 11s 66ms/step - loss: 0.8199 - acc:
0.5015 - val_loss: 0.6882 - val_acc: 0.5416
Epoch 2/10
157/157 [==============================] - 10s 61ms/step - loss: 0.6737 - acc:
0.6379 - val_loss: 0.6736 - val_acc: 0.6246
Epoch 3/10
157/157 [==============================] - 9s 60ms/step - loss: 0.6453 - acc:
0.7419 - val_loss: 0.6468 - val_acc: 0.6458
Epoch 4/10
157/157 [==============================] - 9s 60ms/step - loss: 0.5970 - acc:
0.7941 - val_loss: 0.5612 - val_acc: 0.7774
Epoch 5/10
157/157 [==============================] - 9s 60ms/step - loss: 0.4938 - acc:
0.8330 - val_loss: 0.4497 - val_acc: 0.8254
Epoch 6/10
157/157 [==============================] - 9s 59ms/step - loss: 0.3745 - acc:
0.8661 - val_loss: 0.4164 - val_acc: 0.8388
Epoch 7/10
157/157 [==============================] - 9s 59ms/step - loss: 0.3169 - acc:
0.8856 - val_loss: 0.4039 - val_acc: 0.8540
Epoch 8/10
157/157 [==============================] - 9s 59ms/step - loss: 0.2741 - acc:
0.9035 - val_loss: 0.3969 - val_acc: 0.8608
Epoch 9/10
157/157 [==============================] - 9s 59ms/step - loss: 0.2525 - acc:
0.9095 - val_loss: 0.4198 - val_acc: 0.8620
Epoch 10/10
157/157 [==============================] - 9s 59ms/step - loss: 0.2260 - acc:
0.9236 - val_loss: 0.4186 - val_acc: 0.8692
```

[ ]: