Open Source Software — CSCI-4470 — Spring 2022
Test 2
April 15, 2022

SOLUTIONS

1. (20 points) The code below is the **Word Ladder** example we did in lab. Refer to it to answer the questions below.

```
"""
=====
Words
=====
Words/Ladder Graph
------------------
Generate  an undirected graph over the 5757 5-letter words in the
datafile `words_dat.txt.gz`. This example
is described in Section 1.1 in Knuth's book (see [1]_ and [2]_).
References
----------
.. [1] Donald E. Knuth,
   "The Stanford GraphBase: A Platform for Combinatorial Computing",
   ACM Press, New York, 1993.
.. [2] http://www-cs-faculty.stanford.edu/~knuth/sgb.html
"""
# Authors: Aric Hagberg (hagberg@lanl.gov),
#          Brendt Wohlberg,
#          hughdbrown@yahoo.com

#    Copyright (C) 2004-2019 by
#    Aric Hagberg <hagberg@lanl.gov>
#    Dan Schult <dschult@colgate.edu>
#    Pieter Swart <swart@lanl.gov>
#    All rights reserved.
#    BSD license.

import gzip # 1
from string import ascii_lowercase as lowercase #2
import networkx as nx #3
#-------------------------------------------------------------------
#   The Words/Ladder graph of Section 1.1
#-------------------------------------------------------------------
def generate_graph(words): #4
    G = nx.Graph(name="words") #5
    lookup = dict((c, lowercase.index(c)) for c in lowercase) #6
    def edit_distance_one(word): #7
        for i in range(len(word)): #8
            left, c, right = word[0:i], word[i], word[i + 1:] #9
            j = lookup[c]  # 10
            for cc in lowercase[j + 1:]: #11
                yield left + cc + right #12
    candgen = ((word, cand) for word in sorted(words) #13
```

```
                for cand in edit_distance_one(word) if cand in words) #14
    G.add_nodes_from(words) #15
    for word, cand in candgen: #16
        G.add_edge(word, cand) #17
    return G #18
def words_graph(): #19
    fh = gzip.open('words_dat.txt.gz', 'r') #20
    words = set() #21
    for line in fh.readlines(): #22
        line = line.decode() #23
        if line.startswith('*'): #24
            continue #25
        w = str(line[0:5]) #26
        words.add(w) #27
    return generate_graph(words) #28
if __name__ == '__main__': #28
    G = words_graph() #30
    print("Graph has %d nodes with %d edges"
          % (nx.number_of_nodes(G), nx.number_of_edges(G))) #31
    print("%d connected components" % nx.number_connected_components(G)) #32
    for (source, target) in [('chaos', 'order'),
                             ('nodes', 'graph'),
                             ('pound', 'marks')]: #33
        print("Shortest path between %s and %s is" % (source, target)) #34
        try: #35
            sp = nx.shortest_path(G, source, target) #36
            for n in sp: #37
                print(n) #38
        except nx.NetworkXNoPath: #39
            print("None") #40
```

(a) (2 of 20 points) For the graph, what information is stored in the nodes?

Words

(b) (2 of 20 points) What does an edge between 2 nodes represent?

The words connected by edges differ by one letter.

(c) (2 of 20 points) Which line generates an empty graph?

G = nx.Graph(name="words")

(d) (4 of 20 points) Explain what the following lines of code (lines 13 and 14) do:

```
candgen = ((word, cand) for word in sorted(words) for cand in edit_distance_one(word) if cand
 in words)
```

For each word in the set of all words in the dictionary, it generates all tuples matching the given word with other words in the dictionary such that they only differ by one letter.

(e) (2 of 20 points) Will this code generate the `(word, cand)` pair `('force', 'farce')`"?

No

(f) (2 of 20 points) Will this code generate an edge between `(word, cand)` pair `('force', 'farce')`"? Why or why not?

Yes. Edges are bi-directional and the code will generate the verb—(word, cand)— pair `('farce', 'force')`

(g) (6 of 20 points) Modify the code above to only allow changes to the even numbered character positions. Assume that the first letter of a string (position 0) counts as even. This will prohibit, for example, a path from `force` to `farce`. You do not need to rewrite the entire code above. Just concisely indicate using the line numbers above which lines need to change, what they would change to, and where you would insert any new code.

Change line #8 to read:

```
                    for i in range(0, len(word), 2): #8
```

2. Assume you have a couchDB instance running on `localhost:5984` with administrator `admin` and password `admin`. Also consider the following table of data:

First	Last	Beverage
Wesley	Turner	Coffee
Louie	Dog	Puddle Water
Bugs	Bunny	Carrot Juice

Each question below has a description of the desired operation and the output returned. Provide one or more `curl` command that implement the operation. You may need results from a previous `curl` commend in formulating new commands. You should use the provided output when that is the case.

For example:

Get the welcome message

```
curl http://localhost:5984/
```

Response:

```
{"couchdb":"Welcome","version":"3.2.1","git_sha":"244d428af","uuid":"a3cc56d36b644ee376e45661aeacf43b",\
  "features":["access-ready","partitioned","pluggable-storage-engines",\
  "reshard","scheduler"],"vendor":{"name":"The Apache Software Foundation"}}
```

(a) **(2/20 points)** # Get 3 UUIDs

```
>  curl http://localhost:5984/_uuids?count=3
```

Response:

```
{"uuids":["940012ba002b8d0de8239245980028ea","940012ba002b8d0de823924598002bfb",\
   "940012ba002b8d0de823924598002d2e"]}
```

(b) **(2/20 points)** # Create a drinks database

```
> curl -X PUT http://admin:admin@127.0.0.1:5984/drinks
```

Response:

```
{"ok":true}
```

(c) **(4/20 points)** # Add `Wes` into the database You can assume Louie and Bugs are added as well. The output lines show the return from all three calls in order: `Wes`, `Louie`, `Bugs`.

```
> curl -X PUT \ http://admin:admin@127.0.0.1:5984/drinks/940012ba002b8d0de8239245980028ea \
-d '{"First
   ":"Wes","Last":"Turner", "Beverage":"Coffee"}'

 > curl -X PUT \ http://admin:admin@127.0.0.1:5984/drinks/940012ba002b8d0de823924598002bfb \
 -d '{"First
   ":"Louie","Last":"Dog", "Beverage":"Puddle Water"}' # Not required

> curl -X PUT \ http://admin:admin@127.0.0.1:5984/drinks/940012ba002b8d0de823924598002d2e \
-d '{"First
":"Bugs","Last":"Bunny", "Beverage":"Carrot Juice"}' # Not required
```

Response:

```
{"ok":true,"id":"940012ba002b8d0de8239245980028ea",\
   "rev":"1-ed659d44c0462a25310f95c44351b8ba"}
{"ok":true,"id":"940012ba002b8d0de823924598002bfb",\
   "rev":"1-62cb2c4f3be74e4922a12ed67e9c73ef"}
{"ok":true,"id":"940012ba002b8d0de823924598002d2e",\
   "rev":"1-79af28dadc0a850156201b3842e703f0"}
```

(d) **(4/20 points)** # We are out of coffee, and I refuse to drink `Puddle Water`. Change my document to show I am drinking `Carrot Juice`.

```
curl -X PUT \ http://admin:admin@127.0.0.1:5984/drinks/940012ba002b8d0de8239245980028ea \
-d '{"_rev"
   :"1-ed659d44c0462a25310f95c44351b8ba","First":"Wes","Last":"Turner", "Beverage":"Carrot
    Juice"}'
```

Response:

```
{"ok":true,"id":"940012ba002b8d0de8239245980028ea",\
   "rev":"2-a320f8bd37b1aaddac6ba2765fcf3441"}
```

(e) **(4/20 points)** # Set up an index on `"Beverage"` called `beverage-index` that can be used to quickly search the `drinks` database.

```
curl -X POST admin:admin@localhost:5984/drinks/_index -d '{
"index": {
   "fields": [
   "Beverage"
   ]
},
"name": "beverage-index",
"type": "json"
}' -H 'Content-Type: application/json'
```

Response:

```
{"result":"created","id":"_design/3661a151d6a8590d6e00836a30910faa25247eaf",\
   "name":"beverage-index"}
```

(f) **(4/20 points)** # Now search for everone who drinks `Carrot Juice`.

```
curl -X POST admin:admin@localhost:5984/drinks/_find -d '{
   "selector": {
      "Beverage": {
         "$eq": "Carrot Juice"
      }
   }
}' -H 'Content-Type: application/json'
```

Response:

```
{"docs":[
{"_id":"940012ba002b8d0de8239245980028ea","_rev":"2-a320f8bd37b1aaddac6ba2765fcf3441",\
   "First":"Wes","Last":"Turner","Beverage":"Carrot Juice"},
{"_id":"940012ba002b8d0de823924598002d2e","_rev":"1-79af28dadc0a850156201b3842e703f0",\
   "First":"Bugs","Last":"Bunny","Beverage":"Carrot Juice"}
],
"bookmark": "g1AAAABieJzLYWBgYMpgSmHgKy5JLCrJTq2MT8lPzkzJBYorWJoYG\
BgaJSUaGBglWaQYpKRaGBlbGpmYWloARVKMUkH6OGD6coA6GEHaeJwTi4\
rySxS8SjOTU7OyAOBGGh4"}
```

3. (20 points) Refer to the **Github Action** template for the action yaml file shown and the python file for the `Mystery` class below while answering the following questions.

```yaml
name: Super-Linter
on: [pull_request, push]

jobs:
  super-lint:
    name: Lint code base
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Run Super-Linter
        uses: github/super-linter@v4
        env:
          DEFAULT_BRANCH: master
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

```python
class mystery(object):
    def __init__(this, length, values):
        this.values = [0]*length
        for value in values:
            this.values[value] += 1

    def __str__(this):
        return "length {:d}: {}".format(len(this.values), this.values)

    def __add__(this, value):
        myst = mystery(len(this.values), [])
        myst.values = this.values.copy()
        myst.values[value] += 1
        return myst
```

(a) (2 of 20 points) In the yaml file, which repository contains the command that **runs** the test?
Answer:

`github/super-linter`

(b) (4 of 20 points) What platform and version do the tests run on?
Answer:

ubuntu-latest

(c) (2 of 20 points) One line says which github actions cause the test to run. Rewrite that line so that the test only runs when someone pushes to the repository.
Answer:

on: [push]

(d) (10 of 20 points) Turning to the Python example, create a **Unit Test** file that runs unit tests on the Mystery code. Your test file should test every function in Mystery. All functions should have at least one passing test. Two of the functions have obvious errors. You should provide tests that demonstrate these error cases as well.
Answer:

```
import unittest
import Mystery as M

class TestMystery(unittest.TestCase):
        def test_init_1(self):
            myst = M.mystery(7, [0, 3, 6])
            self.assertEqual( myst.values, [1, 0, 0, 1, 0, 0, 1])

        def test_init_2(self):
            myst = M.mystery(7, [0, 3, 7])
            self.assertEqual( myst.values, [1, 0, 0, 1, 0, 0, 0, 1])

        def test_str(self):
            myst = M.mystery(7, [0, 3, 6])
            self.assertEqual( str(myst), "length 7: [1, 0, 0, 1, 0, 0, 1]")

        def test_add_1(self):
            myst = M.mystery(7, [0, 3, 6])
            self.assertEqual( (myst + 6).values, [1, 0, 0, 1, 0, 0, 2])

        def test_add_2(self):
            myst = M.mystery(7, [0, 3, 6])
            self.assertEqual( (myst + 7).values, [1, 0, 0, 1, 0, 0, 0, 2])

if __name__ == '__main__':
    unittest.main()
```

(e) (2 of 20 points) In our class notes we discussed **White Box**, **Black Box** and **Grey Box** testing. The testing we did in the previous question falls into which of these cattegories?

Answer:

White Box (You looked at the code to determine the correct response and the error conditions)

4. (20 points) We have an issue (https://github.com/rcos/CSCI-4470-OpenSource/issues) in the class issue tracker to **Dockerify** our **Tensorflow** lab. To refresh your memory, here are the lines that we had to execute to fully provision our Docker container so we could use it:

```
> docker run -it -p 8888:8888 -e "DISPLAY"=host.docker.internal:0 tensorflow/tensorflow:latest
> apt-get update
> apt-get install python-tk xterm x11-apps qt5-default
> apt-get install vim
> pip install matplotlib PyQt5
```

Note that this includes adding in an editor. You may also remember that the `Tensorflow` container puts you in the root of file system as the `root` user. We will want to change this.

In your Dockerfile you should:

- provision the above requirements,
- create a directory `/home/tensorflow`,
- create a user `tensorflow`,
- make the user and the directory you created the defaults when you run the image interactively
- finally, we want to run this intractively just like the original image. Make sure you invoke `bash` at the end of the `Dockerfile`

(a) (16/20 points) Write a `Dockerfile` to provision the container described above.

```
FROM tensorflow/tensorflow:latest

RUN apt update
RUN apt install --yes python-tk xterm x11-apps qt5-default
RUN apt install --yes vim
RUN pip install matplotlib PyQt5

RUN useradd -m -s /bin/bash tensorflow
USER tensorflow
WORKDIR /home/tensorflow

CMD ["bash"]
```

(b) (2/20 points) What commands would be used to generate a container named `mytensorflow` under dockerhub account `csci4470` and with version `v1.0`from this Dockerfile, and then push them to Dockerhub? You can assume you are in the same directory as the file and are already logged into the `csci4470` account.

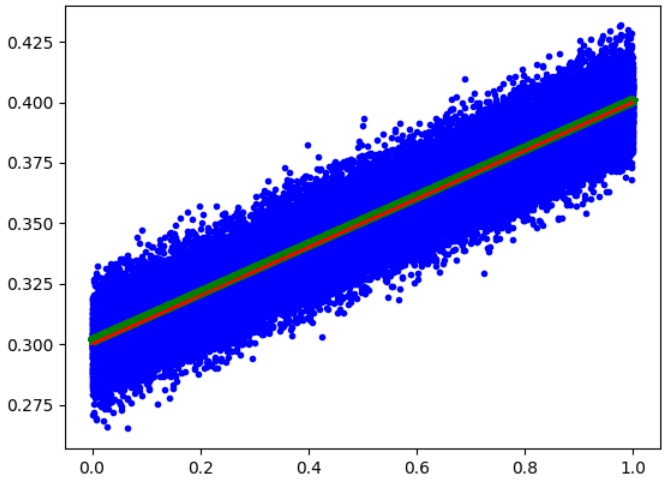
```
docker build -t csci4470/mytensorflow:v1.0 .
docker push csci4470/mytensorflow:v1.0
```

(c) (2/20 points) What command would be used to run the container you just created so that it comes up in interactive mode with a terminal prompt? Be sure to start from the command above and make sure you map the container directory `/home/tensorflow` to the directory `/User/tensorflow/data` on the host machine.

```
docker run -it -p 8888:8888 -v /User/tensorflow/data:/home.tensorflow -e
"DISPLAY"=host.docker.internal:0 csci4470/mytensorflow:v1.0
```

5. (20 points) In class we used Tensorflow utilities to manually find the 2 coefficients m and b of the linear equation $y = m * x + b$. In lab we used Keras to predict types of clothing. For this question, we will use Keras to predict the values of the linear equation we solved in class. This is probably the most difficult problem on the test, but I am providing lots of code. You just need to fill in the blanks.

   The graph below shows a successful run of the completed code. The blue background is a scatterplot of the input data while the double line in the center is the actual line and the prediction from Tensorflow lying overtop of one another.



   Fill in the missing code below. The comments will tell you what you need to do.

   Answer:

```
# MIT License
#
# Copyright (c) 2017 Franois Chollet
#
# Permission is hereby granted, free of charge, to any person obtaining a
# copy of this software and associated documentation files (the "Software"),
# to deal in the Software without restriction, including without limitation
# the rights to use, copy, modify, merge, publish, distribute, sublicense,
# and/or sell copies of the Software, and to permit persons to whom the
# Software is furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
# THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
# DEALINGS IN THE SOFTWARE.

# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
```

```
print(tf.__version__)

def make_noisy_data(m=0.1, b=0.3, n=100):
  x = tf.random.uniform(shape=(n,1,1))
  noise = tf.random.normal(shape=(len(x),1,1), stddev=0.01)
  y = m * x + b + noise
  return x, y

def predict(m, b, x):
  y = m * x + b
  return y

def squared_error(y_pred, y_true):
  return tf.reduce_mean(tf.square(y_pred - y_true))

# Loading data
m = 0.1
b = 0.3
(train_images, train_labels) = make_noisy_data(m=m, b=b, n=60000)
(test_images, test_labels) = make_noisy_data(m=m, b=b, n=10000)


# 1. Create a model with a single neuron in 1 dense layer and 'relu' activation
# 2. Your model should use 'RMSprop' optimization, and mean squared error for
#    both the loss function and the metric
# 3. Train your model on the 'train_images' and 'train_labels' data from above. Use
#    at least three epochs

model = tf.keras.Sequential([
    tf.keras.layers.Dense(1, activation='relu'),
])

model.compile(optimizer='RMSprop',
              loss=tf.keras.losses.MeanSquaredError(),
              metrics=[tf.keras.metrics.MeanSquaredError()])

model.fit(train_images, train_labels, epochs=2)

# 4. Then evaluate the model against the test_images and test_labels
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

#5 Finally, use your model to predict the correct output from the 'test_images'
predictions = model.predict(test_images)

# Calculate the actual values
actual = predict(m, b, test_images)

# Report
print('\nTest accuracy: ', test_acc, '\nTest Loss: ', test_loss)
print("Accuracy (MeanSquaredError): ", squared_error(predictions, actual).numpy())

# Plot
plt.plot(np.squeeze(train_images), np.squeeze(train_labels), 'b.')
plt.plot(np.squeeze(test_images), np.squeeze(predictions), 'g*')
plt.plot(np.squeeze(test_images), np.squeeze(actual), 'r-')
plt.show()
```

---

model =

```
# 4. Then evaluate the model against the test_images and test_labels

# WRITE YOUR CODE BELOW. (4/20 points):

# test_loss, test_acc =
```

test_loss, test_acc =

```
#5 Finally, use your model to predict the correct output from the 'test_images'.

# NOTE: YOU DO NOT NEED softmax FOR THIS

# WRITE YOUR CODE BELOW. (4/20 points):

# predictions =
```

predictions =

```
# Calculate the actual values
actual = predict(m, b, test_images)

# Report
print('\nTest accuracy: ', test_acc, '\nTest Loss: ', test_loss)
print("Accuracy (MeanSquaredError): ", squared_error(predictions, actual).numpy())

# Plot
plt.plot(np.squeeze(train_images), np.squeeze(train_labels), 'b.')
plt.plot(np.squeeze(test_images), np.squeeze(predictions), 'g*')
plt.plot(np.squeeze(test_images), np.squeeze(actual), 'r.')
plt.show()
```

Use this page for scratch.

Use this page for scratch.

Use this page for scratch.

Use this page for scratch.