Insert Assignment Title Here
02807 Computational Tools for Big Data

Anonymous authors

Insert hand in date here

1 Exercise 1.1

The following pipeline:

1. Deletes all punctuation, commas and quotes from file
2. Translates whitespace to newline
3. Sorts it
4. Counts occurrence of each word
5. Sorts it numerically in reverse (largest number first)
6. Prints the top 10 lines

```
tr -d ",.'" < test | tr ' ' '\n' | sort | uniq -c | sort -n -r | head -n 10
```

2 Exercise 1.2

The following unix script deletes all lines that contains a number with 5 or more digits

```
sed "/[0-9]\{5,\}/d" < test2
```

3 Exercise 1.3

The following pipeline:

1. Translates all tabs into spaces in the shakespeare.txt file
2. Removes all characters satisfying `[ ^ a-zA-Z ]`
3. Translates all spaces to newlines
4. Translates upper case to lower case
5. Sorts the lines
6. Keeps only unique lines
7. Uses dict file as plain string to match on the entire individual lines and print only the lines that don't match anything in dict.
8. counts the lines i.e. the misspelled words.

```
tr '\t' ' ' < shakespeare.txt | sed 's/[^a-zA-Z ]//g' | tr ' ' '\n' | tr A-Z a-z |
   sort | uniq | grep -F -x -v -f dict | wc -l
```

4 Exercise 1.4

We chose to use gbar instead of AWS.

5 Exercise 1.5

Git pull on gbar:

```
gbarlogin1(s152165) $ git pull
remote: Counting objects: 15, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 15 (delta 4), reused 1 (delta 1), pack-reused 2
Unpacking objects: 100% (15/15), done.
From https://github.com/ttsoftware/computational-tools
   1d522e1..407cabb  master     -> origin/master
Merge made by recursive.
 ENyYffaq.txt |   40 ++++++++++++++++++++++++++++++++++++++++
 comm         |    1 +
 matrix3      |    3 +++
 week2.py     |   52 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 week3.py     |   15 +++++++++++++++
 5 files changed, 111 insertions(+), 0 deletions(-)
 create mode 100644 ENyYffaq.txt
 create mode 100644 comm
 create mode 100644 matrix3
 create mode 100644 week2.py
 create mode 100644 week3.py
~/computational-tools
```

For a reference of commits see https://github.com/ttsoftware/computational-tools/commits/master.

6 Exercise 2.1

```
def read_matrix(filename):
    """
    Reads a file and tries to construct a matrix from data in the file

    :param filename: name of file it be read
    :return: list of lists of the numbers in the file
    """
    f = open(filename, 'r')

    # Reads all lines from file into list
    lines = f.readlines()

    matrix_list = []
    for line in lines:
        # Using list comprehension to construct a new list of the numbers
        # contained in each line
        matrix_list.append([float(c) for c in re.split("[, ]", line.rstrip())])
    f.close()
    return matrix_list
```

The `read_matrix` method, when run with the file mentioned in the excercise, returns the following:

```
[[0.0, 1.0, 1.0, 3.0, 0.0], [0.0, 2.0, 3.0, 4.0, 10.0], [8.0, 2.0, 2.0, 0.0, 7.0]]
```

```
def write_matrix(array, filename):
    """
    Interprets a matrix (list of lists) and writes it to filename

    :param array: Matrix (list of lists) to be written to file
    :param filename: Name of the file to write to
    """
    f = open(filename, 'w')
    output = ""
    for l in array:
        for c in l:
```

```
            output += str(c) + ' '
        output = output[:-1] + '\n'
    f.write(output)
    f.close()
```

The `write_matrix` method, when run with a random matrix, creates the following file output:

```
2 3 5
1 4 2
```

7 Exercise 2.2

```
def bit_strings(N):
    """
    Takes a number N and constructs all bit permutations of size N

    :param N: Size of permutations
    :return: List of all permutations of bit strings of size N
    """
    pools = [[0, 1]] * N
    result = [[]]
    for pool in pools:
        result = [x+[y] for x in result for y in pool]
    return result
```

For $N = 3$, this method yields the following result:

```
[[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 1]]
```

8 Exercise 2.3

```
def bag_of_words(filename):
    """
    Reads a very specific json file and constructs a bag-of-words representation
    of people's comments and their ability to get free pizza.

    :param filename: Name of file to be read
    :return: Bag-of-words representation
    """
    f = open(filename)
    lines = f.readlines()
    request_texts = []
    theD = {}
    cnt = 0

    # Regular expression that matches request_text line
    pat = re.compile('\s+"request_text": "(?P<text>.+)"')
    # Regular expression that matches requester_received_pizza line
    patr = re.compile('\s+"requester_received_pizza": (?P<pizza>.+),')
    results = []
    for line in lines:
        # If line is matches request_text
        result = re.search(pat, line)
        if result:
            # Saves the request_text for later use
            request_texts.append(result.group('text').lower())

            # Iterate over all words
            for word in re.findall("[\w\']+", result.group('text')):
                word = word.lower()

                # Put words not already found into dictionary
                if word not in theD.keys():
                    theD[word] = cnt
                    cnt += 1

        # Make list of all the results of pizze beggars.
        pizza_result = re.search(patr, line)
```

```
        if pizza_result:
            results.append(0) if pizza_result.group('pizza') == 'false' else results.append(1↩
                )

    # Use word dictionary to create bag-of-words
    bag = []
    for i, text in enumerate(request_texts):
        vec = [0] * len(theD) + [results[i]]
        for word in re.findall("[\w\']+", text):
            vec[theD[word]] += 1
        bag.append(vec)

    return bag
```

9 Exercise 3.1

[[-5.09090909] [1.18181818] [2.24242424]]

10 Exercise 3.2

-1.43463628748

11 Exercise 3.3

11.1 Top five

movie id	rating count
2858	14800
260	13321
1196	12836
1210	11598
2028	11507

11.2 Active title

movie id	rating count
1	8613
2	2244
3	1442
4	464
5	890
6	3646
7	1562
9	271
10	3144
11	3919
12	378
13	323
14	542
15	359
16	2587
17	3363
18	524
19	965
20	406
21	4914

```
22                1266
23                 360
24                1984
25                3578
26                 353
28                 726
29                1637
30                 270
31                 439
32                5962
34                6814
36                3673
39                4935
41                 958
42                 634
43                 572
44                 867
45                1863
46                 516
47                4669
48                1137
50                8054
52                1569
57                 337
58                2051
60                1147
62                2000
63                 317
65                 295
69                1166
70                2885
72                 338
73                 855
74                 357
76                 508
79                 297
81                 525
82                 345
85                 660
86                 801
                   ...
[2161 rows x 1 columns]
```

11.3 Top 3 female ratings

```
movie id  mean
745       4.644444
1148      4.588235
3022      4.575758
[3 rows x 1 columns]
```

11.4 Top 3 male ratings

```
movie id  mean
2905      4.639344
858       4.583333
2019      4.576628
```

```
[3 rows x 1 columns]
```

11.5 Top 10 female difference

```
      movie id    mean\_x    mean\_y  mean\_diff
1129      2084  3.861111  2.666667   1.194444
13          15  3.200000  2.341270   0.858730
579       1088  3.790378  2.959596   0.830782
864       1592  3.057143  2.233766   0.823377
924       1707  2.486486  1.683761   0.802726
818       1460  3.156250  2.435484   0.720766
116        203  3.486842  2.795276   0.691567
1382      2468  3.254717  2.578358   0.676359
299        506  3.862745  3.190476   0.672269
373        650  3.800000  3.136364   0.663636
[10 rows x 4 columns]
```

11.6 Top 10 male difference

```
      movie id    mean\_x    mean\_y  mean\_diff
2019      3658  2.900000  3.779661  -0.879661
1934      3487  3.000000  3.765625  -0.765625
1315      2377  2.250000  2.994152  -0.744152
781       1382  2.100000  2.837607  -0.737607
1696      3036  2.578947  3.309677  -0.730730
638       1201  3.494949  4.221300  -0.726351
298        504  2.300000  2.994048  -0.694048
2079      3760  2.878788  3.555147  -0.676359
1194      2165  2.888889  3.536585  -0.647696
1713      3066  3.090909  3.737705  -0.646796
[10 rows x 4 columns]
```

11.7 Top 5 standard deviation

```
movie id  std
1924      1.455998
2314      1.372813
3864      1.364700
2459      1.332448
231       1.321333
[5 rows x 1 columns]
```

12 Exercise 3.4

13 Exercise 3.5

`python week3_cython.pyx It took 0.771023988724 seconds to compute the sum 500 times.`

`python week3_cython_run.py It took 0.586192846298 seconds to compute the sum 500 times.`

We notice the reduce the running time by approximately 0.18 seconds or approximately 20%.

14 Exercise 4.1