DATA SCIENCE AND MACHINE LEARNING (MSC)

DAMA60: Algorithmic Techniques and Systems for Data Science and Machine Learning

Academic Year: 2023–2024

#2 Written Assignment		
Submission Deadline	**Wednesday, 10/1/2024 23:59 Athens Time**	
Name / Student Id	**Kritikos Antonios**	**std157978**

Remarks

The deadline is definitive.

An indicative solution will be posted online along with the return of the graded assignments.

The assignment is due via the STUDY submission system. **You are expected to turn in a document (.DOC, .ODT, .PDF) and a file containing a Python program:**

- 1 document file (this document) with the answers to all the questions, along with the Python code snippets for Topic 5.
- 1 file with the complete Python program that answers Topic 5.

You should not make any changes in the written assignment file other than providing your own answers. You should also type all of your answers into Word and not attach any handwritten notes as pictures into your work. Make sure to name all the files (DOC file and Python program file) with **your last name first followed by a dash symbol and the names of each component at the end**. For example, for the student with last name Aggelou the files should be named as follows: Aggelou-HW2.doc (or Aggelou-HW2.pdf), Aggelou-Topic5.py. **Also, please include comments before each command to explain the functionality of the command that follows.**

Topic	Points	Grades
1. Online Quiz	40	
2. MapReduce	15	
3. Locality Sensitive Hashing	15	
4. Data Streams	15	
5. Python	15	
TOTAL	**100**	**/100**

Topic 1: Online Quiz

(40 points) Complete the corresponding online quiz available at:

https://study.eap.gr/mod/quiz/view.php?id=26732

You have **one effort** and unlimited time to complete the quiz, up to the submission deadline.

Topic 2: MapReduce

(15 total points) We have the following two tables **r** (on the left) and **s** (on the right) and we wish to apply the NATURAL JOIN operator on them.

r			
A	**B**	C	**D**
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

s		
B	**D**	E
1	a	α
3	a	β

(a) **(7 points)** Fill in **the missing values** ("**?**") of the intermediate key-value pairs produced from table r.

Answer:

(Key,	**Value)**
((B, D),	(r, (A, C)))
((**1**, a),	(r, (α, α)))
((2, **a**),	(r, (β, γ)))
((4, b),	(r, (**γ**, **β**)))
((1, a),	(**r**, (**α**, **γ**)))
((2, b),	(r, (δ, β)))

(b) **(3 points)** Fill in **the missing values** ("**?**") of the intermediate key-value pairs produced from table s.

Answer:

(Key,	**Value)**
((B, D),	(s, E))
((1, **a**),	(**s**, α))
((3, a),	(s, **β**))

(c) **(5 points)** Fill in **the missing values** ("**?**") for the final key-value pairs produced by the reducer.

Answer:

(Key,	**Value)**
((**1**, **a**),	(α, α, α))
((1, a),	(**α**, **α**, **γ**))

Topic 3: Locality Sensitive Hashing

(15 total points) We have the following table representing the elements of sets S_1, S_2, S_3, S_4 from the universal set {1, …, 10}.

	S1	S2	S3	S4
1	0	1	1	0
2	1	0	0	0
3	0	0	1	0
4	0	1	0	0
5	0	1	1	0
6	1	0	0	1
7	1	0	1	1
8	0	1	0	1
9	0	1	1	1
10	0	1	0	1

We want to use Minhashing to approximate the similarity among these sets. Assume three hash functions H_1, H_2, H_3 that permute the original row ordering as follows:

H_1: 6, 2, 8, 1, 4, 3, 7, 5, 10, 9

H_2: 2, 3, 5, 8, 1, 4, 6, 10, 9, 7

H_3: 3, 5, 6, 4, 2, 9, 8, 1, 10, 7

(a) (**7 points**) Fill in **the missing values** ("**?**") in the following signature matrix considering original row numbering.

Answer:

	S_1	S_2	S_3	S_4
H_1	**2**	1	**4**	**3**
H_2	3	**1**	1	**4**
H_3	**5**	1	**2**	1

(b) (**8 points**) Fill in **the missing values** ("**?**") in the following table for Jaccard similarities of input sets and minhash signatures.

Answer:

	Initial Sets Jaccard	**Signatures Jaccard**
S1,S2	**0**	0
S1,S3	**1/7**	**0**
S1,S4	2/6	**0**
S2,S3	**3/8**	**1/3**
S2,S4	**3/8**	1/3
S3,S4	2/8	**0**

Topic 4: Data Streams

(15 total points) Consider the elements in the sequence <17, 20, 0, 8, 12> that come as a stream, and the following two hash functions, $f_1(x) = (3x+5)$ MOD 32 and $f_2(x) = (2x+4)$ MOD 32. We will be using the Flajolet-Martin algorithm to estimate the number of distinct elements in the sequence.

(a) (**4 points**) First, we must hash each stream element using each one of the hash functions and use a binary representation for each hash function result for each element. Fill in **the missing values** ("**?**") in the following table.

Answer:

Stream Element (x)	Binary Representation of $f_1(x)$	Binary Representation of $f_2(x)$
17	011000	000110
20	**000001**	**001100**
8	**011101**	**010100**

(b) (**4 points**) Using the Flajolet-Martin algorithm, estimate the number of distinct elements using $f_1(x)$ after having seen stream element 12.

Answer:

The number of elements is: $N_1 = 2^{3}$

(c) (**4 points**) Using the Flajolet-Martin algorithm, estimate the number of distinct elements using $f_2(x)$ after having seen stream element 12.

Answer:

The number of elements is: $N_2 = 2^{2}$

(d) **(3 points)** How can we use both N_1 and N_2 to find a more accurate estimate of the number of distinct elements in the stream?

Answer:

A more accurate representation is provided by **the average of** the two values: $N = (N_1 + N_2) / 2$

Topic 5: Python

(15 total points) In this exercise you will use libraries of Python to hash words into Bloom Filters. Data is provided in the text file "`WA2_5.txt`". You will read the file, split the sentences into word tokens and insert the tokens into a Bloom filter. Then, you will check another corpus of words against the Bloom Filter.

Here, we will construct a Bloom Filter using the error rate and the maximum number of elements to be inserted in the filter. The optimal number of bits per element in a Bloom Filter given an error rate **ε**, having **n** bits and **m** elements is:

$$\frac{n}{m} = -\frac{log_2 \varepsilon}{ln2} \approx -1.44 log_2 \varepsilon$$

This means that for a given false positive probability **ε**, the length of a Bloom filter **n** is proportionate to the number of elements being filtered **m**.

A Bloom filter library which is called Bloom-filter2 is required. More information about this library can be found here: https://github.com/remram44/python-bloom-filter

(a) (**4 points**) Complete the following Python code so as to construct a Bloom filter in Python, using the bloom_filter2 package. The Bloom filter should host a maximum of 50 elements and exhibit an error rate of 0.5. Insert the elements of the text file "WA2_5.txt" into the Bloom filter.

Incomplete Python Code:

```
from bloom_filter2 import BloomFilter
import math

max_elements = 50
error_rate=0.5

all_tokens = []

b = BloomFilter(max_elements=max_elements, error_rate=error_rate)

with open("WA2_5.txt") as my_file:
    for line in my_file:
        line_tokens = line.split()                    # split the line to get tokens
        tokens = [l.upper() for l in line_tokens]     # convert each token to uppercase
        all_tokens.extend(tokens)                     # maintain a list with all tokens seen
        for t in tokens:                              # add each of these tokens to the BF
            b.add(t)
```

(b) (**2 points**) Using the formula above, calculate and output the size of the Bloom filter.

Incomplete Python Code:

```
# calculate and print the optimal size of the BF using
# the aforementioned formula for the existing configuration:
size = max_elements * (-1.44 * math.log(error_rate, 2))
print(size)
```

(**c**) (**3 points**) Using the Bloom filter created in (a), check for membership the words of the following list: words = ["THE","BE","TO","OF","AND","A","IN","THAT","HAVE","I"] and count the True Positives, the False Positives, and the True Negatives.

Incomplete Python Code:

```
# The list of words to check in the BF
words = ["THE","BE","TO","OF","AND","A","IN","THAT","HAVE","I"]

TP = 0
FP = 0
TN = 0

# Check each word in the list of words for membership in the BF
# and characterize the result accordingly.
for w in words:
    if w in b and w in all_tokens:
        TP += 1
    elif w in b and w not in all_tokens:
        FP += 1
    else:
        TN += 1

print("TP=",TP, "FP=",FP, "TN=",TN)
```

(**d**) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Answer:

The code in your .py file with name **Kritikos-Topic5.py** conforms to the module's directions, and opens and displays correctly in IDLE.

(**e**) (**3 points**) Follow the guidelines on the first page of this assignment and incorporate the .py file that answers this topic into your submission.

Answer:

Your code executes correctly when run from your .py file with name **Kritikos-Topic5.py** using IDLE.