Mining massive Datasets WS 2017/18

Problem Set 10

Rudolf Chrispens, Marvin Klaus, Daniela Schacherer

January 15, 2018

Exercise 01

Let the input to the hash functions and thus the elements in S be binary strings s. Then possible hash functions for a bloom filter could be

- h1 takes every third position of s starting from position 0, treats them as a number and computes modulo 11
- h2 takes every third position of s starting from position 1, treats them as a number and computes modulo 11
- h1 takes every third position of s starting from position 2, treats them as a number and computes modulo 11

These hash functions are independent from each other as they use different elements of s for computing the hash value.

Exercise 02

a) The probability that a random element ($m = 1$) gets hashed to a given bit in the bit array with $n = 5$ can be computed by the following formula:

$$1 - (1 - \frac{1}{n})^{n(m/n)} = 1 - e^{-m/n} = 1 - e^{-1/5} = 0.1813$$

For $h_1(x)$ each bit is equally likely to be hit. For $h_2(x)$ this is not the case as $2x + 3$ always results in an odd number which will then be taken modulo 5.
Bit array state: | 1 | 0 | 0 | 0 | 1 |

b) With $k = 2, n = 5$ and $m = 1$ unknown the probability for false positives is:

$$(1 - e^{-km/n})^k = (1 - e^{-2/5})^2 = 0.109$$

Exercise 03

With the same formula as in Exercise 2b we receive with $n = 8$ billion, $m = 1$ billion and $k = 3$

$$(1 - e^{-km/n})^k = (1 - e^{-3*1/8})^3 = 0.0306$$

with $k = 4$ we get

$$(1 - e^{-4*1/8})^4 = 0.024$$

Exercise 04

a) The keys for this problem should be *(university, course)*. So we geht $\frac{1}{20}$ of all courses. Now we could group by this keys and count up the number of students. For every key *(university, course)* we get the number of students. If I get this question right, we next have to group the universitys together and divide all the summed up students by the number of courses we had for this university.

b) The key for this problem should just be *students*. After taken $\frac{1}{20}$ of all student we first count them to compare that later and then filter them by the grades (higher than 2.0). The number of student left, gives us the proportion to the original number of students.

c) For this problem the key should be *(university, course, student)*. We just need the university in the key if we want to distinguish between courses with the same name in different universitys. First we count for every key the number of students to compare that later. If we know filter the grades of this students (better than 1.7), we group them by the course and check if the number of students left is more than a half of the original number of students in that class.

Exercise 05

a) This command generates the following output:

```
('step', 1)
('mankind.', 1)
('giant', 1)
('man,', 1)
('a', 2)
('for', 2)
('A', 1)
('small', 1)
('leap', 1)
```

Table 1 – Example Completeness

price-series-name	timestamp-gap-start	timestamp-gap-end
t2.micro___Linux-...	2017-11-29 02:00:26	2017-11-28 02:00:06

I think this is pretty much what I expected. Every work is counted correctly. The punctuation mark belong to a word, if it is that what you asked for? Maybe if we output with *pprint()*, I want the output already sorted. But except of this there is nothing special.

b) I don't know if this just happen because my laptop is too slow. But the output of the data is delayed for one second while the application count the words. The output varies from time to time but here is an example:

```
('thx', 446792)
('k', 446792)
('bai', 893584)
```

The *yes* command is pretty fast. The string was computed 446792 times. With blanks the string have 13 characters what equals 13 bytes. Since $13 * 446792 = 5808296$, the throughput is *5808296 b/s = 5808.296 kB/s = 46.466368 Mb/s.*

Exercise 06

```
a)
see function "checkCoherency" in ex10_6.py
b)
see function "checkCompleteness" in ex10_6.py
c) timeseries related to prices- ca-central
checkCoherency
number of found inconsistencies:  27
checkCompleteness
number of found inconsistencies:  37 (bei > time) 39 ( bei >= time)
```