INTRODUCTION TO BIG DATA PROJECT

TERM: FALL SEMESTER 2017

Theodoros Kalmanidis ID:218580

In this project we have collected a sample of posts in a social network and we assume that the data flow as a stream and that the volume is such that cannot be stored in a database. So we have to do a stream processing which means to analyze our data by only gathering a small sample as the data stream grows bigger.
The data are in a json format, and contain twitter post information from the well known Twitter application. Of course we have pre-processed our data so that we can manipulate them through python language. We can very easily store our json stream file as a dataframe with the help of the json library which transforms any json file to a more human readable format.
In this particular project we would use sampling and sketching algorithms to process our data which we will describe later.

Q1:Streaming processing: Data Sketching – 25%

You are asked to **count** the number of times each tag appears in the stream

1) The first approach would be to have a counter for each tag. How much space does this take?Implement it in Python, visualize the results and save them in file entitle 'tag-counter' as a csv file.

   For practical reason we will show the top 6 tags but of course someone can see all the tags and their counters in the produced csv file

Crimea	**2714**
Putin	1177
Russia	1609
Ukraine	2164
Ipad	1149
Usa	1582
Table 1	

And in the image above we can see visually the top 6 tags:

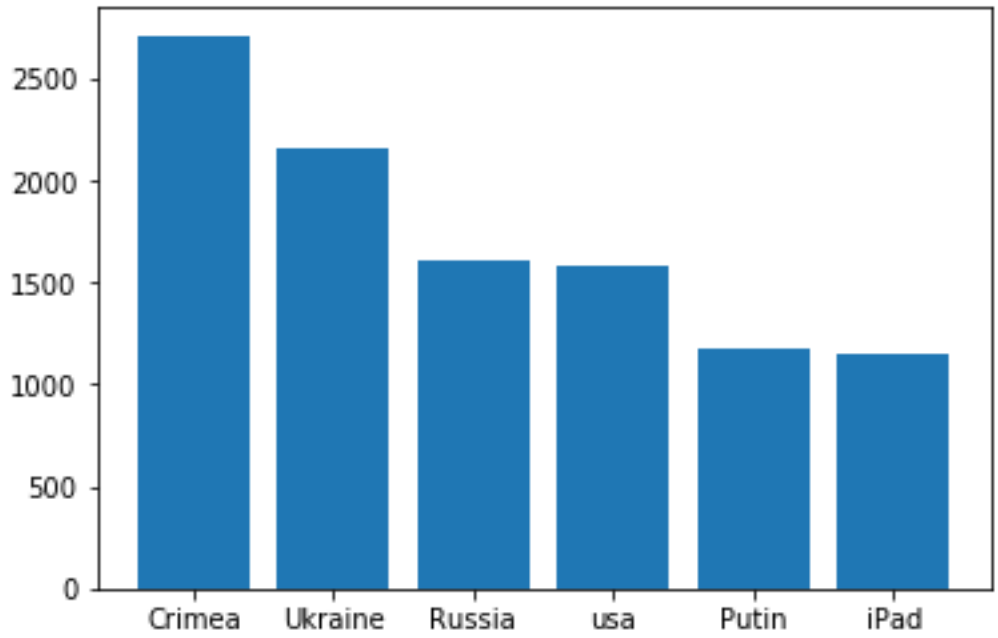


Image 1

	A	B
1		0
2	1000worlds	2
3	13WHAM	1
4	15ofmarch	1
5	1920s	1
6	1950s	1
7	1960s	1
8	19thcentury	1
9	1ACCA	1
10	1TV	1
11	1YearWithoutMCR	1
12	1u	2
13	200aday	1
14	23MarchTwitterGT	1
15	23street	1
16	2A	22
17	2a	2
18	30SecondsToMars	1
19	30secondstomars	2
20	30secondstomarsinn	1
21	30secondstomarsinr	1
22	30stm	1
23	35headsofstate	4
24	370Qs	1
25	3D	1
26	3dprinter	4
27	3dprinting	1
28	3yeses	2
29	401k	1
30	4Jobs	1
31	4NewIran	1
32	4thingstoknow	2
33	4thofJuly	1
34	5000movies	1
35	5SOSTOOHIO	1
36	5Things	3
37	5countries5tees	1
38	702brain	5
39	70s	2
40	80sMusic	1
41	80slifestyle	1

tag-counter

Image 2

This is a small idea about what the counter-tag.csv file contains. We can see with alphabetical order all the tags paired with their counters

2) An approximate solution, that takes much less space, is to use the *count-min sketch* (you can find code for that in the Web)
   a) Discuss the advantages & disadvantages of such an approachin the current problem

The frequency **count–min sketch** is a probabilistic data structure that serves as a frequency table of events in a stream of data. It uses hash functions to map events to frequencies, but unlike a hash table uses only sub-linear space, at the expense of overcounting some events due to collisions.
The count min sketch is practically a 2-D array with size of WxD. One advantage is that is storage space is adjustable which means the number of rows and columns someone chooses to create the sketch will affect how big or small the array will become. Each row (w) uses a hash function to map the different inputs of the array, so every input is automatically hashed in all rows. If the same input arrives multiple times the hash functions increase the counter for that specific input item.

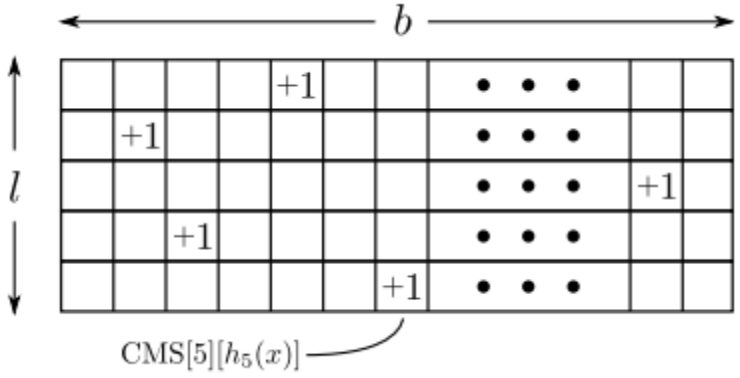


Image 3

As we see in the image above the sketch store bits, and that the trick of the algorithm we don't store distinct elements as we would do with a usual counter but we just store the counters. Each entry of the array is initially zero. hash functions are chosen uniformly at random from a pairwise independent family.
The speed of the performed sketching is high because we only increase the counter when an update is needed. When the elements are low frequent the relative error get higher but usually the Count Min sketch is implemented in high frequent elements in large datasets and produces a good estimation. The CMS only overestimated and never underestimate the true count in large datasets
When we want to do the opposite, to ask the sketch to give us the counter for a specific stored item we get D outputs as the rows. From the D outputs we choose the smaller in value, in newer versions the algorithm is clever enough to give us immediately the smallest one.
The reason we choose the smallest value is because in that way we are taking the value with the least number of hash collisions which is one of the main disadvantages of the algorithm. Of course if we have more columns and rows we decrease the probability of collisions inside the array and we increase the accuracy of the output.

accuracy ϵ	certainty δ	#cols w	#rows d	#cells wd
0.1	0.1	28	3	84
0.1	0.001	28	7	196
0.01	0.001	272	7	1904
0.001	0.001	2719	7	19033

Image 4

In our count min sketch implementation we choose 272 columns and 7 rows because as we can see from the image above we achieve pretty good accuracy and certainty for our accuracy

b) Implement it and evaluate its accuracy. The comparison can be done based on real frequency of each counter which is known.

After the installation of the count-min package we add all the tags of the stream inside count min sketch.

```
('Crimea', 2714)
('Ukraine', 2164)
('Russia', 1609)
('usa', 1582)
('Putin', 1177)
('iPad', 1149)
('anal', 1148)
('milf', 1148)
('xxx', 1146)
('USA', 702)
```

```
In [192]: runfile('C:/Users/teo/cms.py', wdir='C:/Users/teo')
2769
2212
1861
1686
1222
37
1201
1179
1303
757
```

Image 5

We can observe that eight out of ten tags have been accurately computed by the sketch. Are the two values wrong that predicted wrong? The answer is that we choose many rows but only 7 buckets (d rows) so due to collisions there is high chance some counter may be wrong. Comparing those results with the real we can see that for the most values we get accuracy very close to 10%.

Q2: Streaming processing: Sliding Window – 25%

Sliding windows: 15%

Maintain a sample of the data that appear in the stream. The sample should have a small size relative to the size of the data set. Design a *priority sampling* window and a *reservoir* sampling window

Use the above approaches to the following questions (as applicable to every time window)

1. Who are the top-10 most active user?
2. Who are the least active users?

Real Values		Priority Sampling(100)		Reservoir Sampling(100)		Priority Sampling(3100)		Reservoir Sampling(3100)	
User_ID	Posts	User_ID	Posts	User_ID	Posts	User_ID	Posts	User_ID	Posts
2415336255	233	79502144	2	473833548	2	2415336255	35	2415336255	91
36760934	101	2358643152	2	1275624552	2	36760934	26	2149641106	81
473833548	60	1850157565	1	1958450166	2	2347920332	15	204448003	15
1958450166	60	397186854	1	2350733744	2	2316586249	8	97596249	13
2316586249	56	62200005	1	1445691278	2	1868900130	8	1454416338	11
2316444259	56	1184706728	1	2153483660	1	415826305	8	554692057	10
2354604882	55	1683128491	1	245451401	1	2351413850	7	1852547324	9
2318931576	55	346813942	1	1215365448	1	2343042781	7	2174512140	9
1868900130	54	2182819387	1	175648953	1	2314988388	7	2314805664	8
97596249	54	726147038	1	1680427825	1	2354604882	7	2316444259	8
339223177	3	427402721	1	1595289919	1	2316595494	7	169443326	8
261508113	3	619368546	1	2379923131	1	2344390543	6	993649740	8
224246266	3	579494646	1	188368109	1	425061448	6	2318931576	7
1535603702	3	69774449	1	188368109	1	2314735766	6	1448250524	7
30893405	3	1401624636	1	1241069040	1	1708639452	1	2342626099	5
272693437	3	15644221	1	1013523828	1	101102896	1	258230465	5
2357996383	2	2355651788	1	232382227	1	1310945894	1	2188272499	5
192477353	2	494009898	1	174656965	1	1610906258	1	2358818136	5
2355822002	1	340875095	1	273465530	1	925504734	1	57811254	4
249427906	1	1375371229	1	539684179	1	1964661985	1	2403254339	4
109241118	1	726147038	1	1849605206	1	25181908	1	19003991	4
1166929328	1	1948396310	1	932542680	1	312551423	1	1919637056	4
32607660	1	2338800001	1	433137920	1	393382936	1	475782561	4

Table 2

In our experiment we tried sampling with 100 and 3100 (which is the 10% of the whole dataset). What we observe is that reservoir sampling in the 100 and 3100 item experiment catched more “active” users than the priority. Of course in the whole dataset over 90% of

the post are users who don't post so often, so it makes sense to see so low post value in the 100 item experiment. But what we can figure out is that reservoir sampling was more accurate due to the fact that the priority sampling neglects the expired elements where as the reservoir maintains them with its fixed-size uniform sampling method.

3. Which are the top-10most popular tags?

Crimea	**10**
Ukraine	7
usa	6
xxx	5
anal	5
milf	5

Table 3

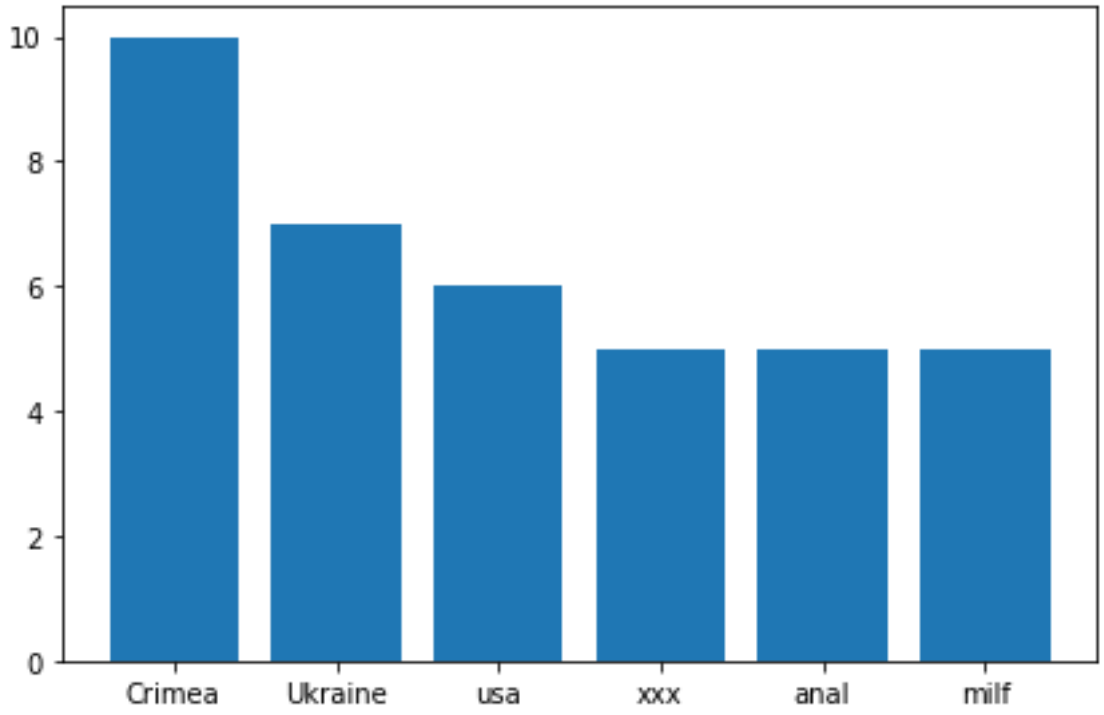


Image 6 Priority sampling

Ukraine	**9**
Crimea	7
Russia	7
Usa	6
Eur	3
Uk	3

Table 4

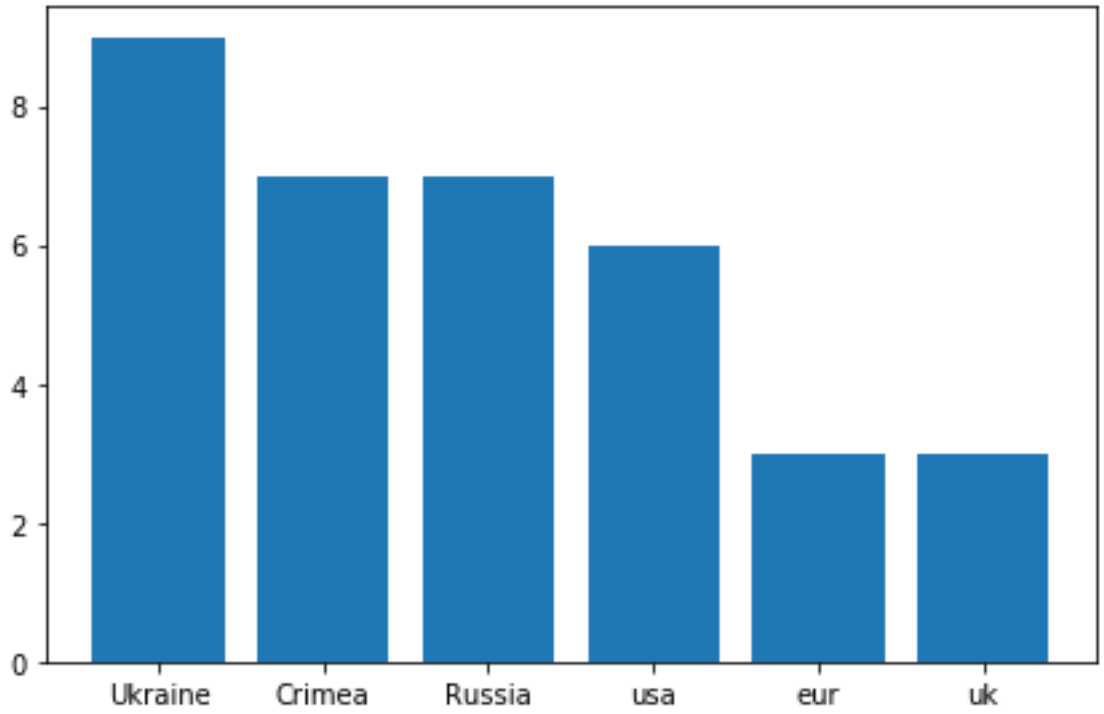


Image 7 Reservoir sampling

4. How can you compare the two sampling methods?

We can compare them in terms of space and speed. Priority sample is a little bit slower because we have to compute the priorities of all items and sort them in decreasing priority order. Its advantage is that even though we don't discard the items with low priorities we can see the whole stream even we can't store it, but can store the items with highest priority. Basically Priority sampling is a non uniform sampling, so the costs is that we have to loop all the stream to pick up the "best" sampling probabilities.
So the two costs are Estimation Variance and Expected Sample size scan, but we have better precision in our estimations.
In reservoir sampling we have a fixed size k uniform sample from an N size stream in one pass, which means that we don't need to know the stream size, we just include the first k items and then when more items arrive we choose those with probability p(n) = k/n, n > k, if the item is not saved we discard it where as in priority we keep it

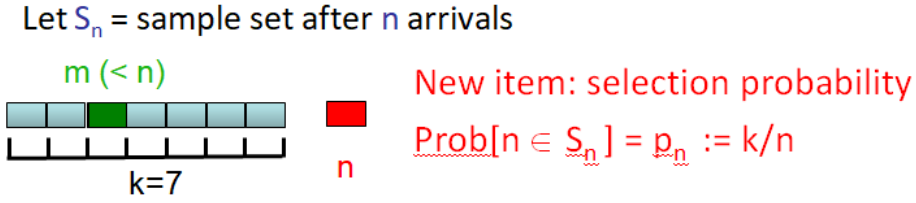


Image 8
In the image below we can see a disadvantage of the reservoir sampling, which is that when the weights are uneven , the rejection probability changes especially when the largest weight differs a lot from the average. As we can see e if the first eight boxes above where %1 full and the last 100% full. It's going to reject roughly 80% of the time. It can of course be much worse when n is larger.

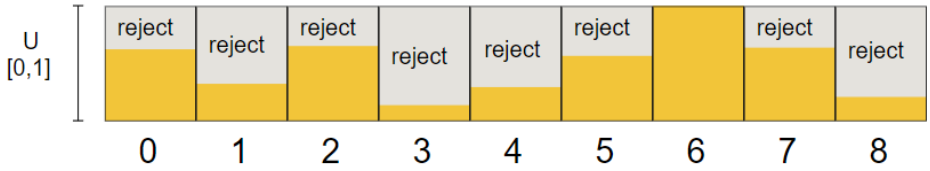


Image 9
So for the reservoir sampling is more difficult to find uniform samples and for priority is more difficult to maintain as the stream grows. To sum up:
Priority Sampling

- Works well when we need to consider "timeliness" of the data
- Data is considered to be expired after a certain time interval
- "Sliding window" in essence is such a random sample of fixed size (say k) "moving" over the most recent elements in the data stream

Reservoir Sampling

- Works well when we have only inserts into a sample
- The first element in the data stream can be retained in the final sample
- It does not consider the expiry of any record

Counting Unique: 10%

You are asked to count the number of unique tags:

a. What is the simplest approach you can think of?

The simplest approach is to save all tags in a list, then remove the dublicates using the set function and then print the length of the list without the dublicates.

b. What could an approximate approach be that saves space and/or time?

An approximate approach could be to use the hyperloglog algorithm. But why hyperloglog ? First of all it uses memory very efficiently and it can estimate large cardinalities (over one billion) with pretty good accuracy. By using hyperloglog algorithm we use a spherical averaging scheme in addition to what was used before.We remove sorting operations that previous algorithms where using (loglog,superloglog) that why we see so much gain in memory performance even if the data are too big (practically hyperloglog can count one billion distinct items with an accuracy of 2% using only 1.5 KB of memory.

In the image below we can see the size of a hyperloglog data structure versus two other similar probabilistic data structures

Size estimates for the number of unique words in Wikipedia

	Elements	Relative error	Processing time[a]	Structure size[b]
Morris counter[c]	1,073,741,824	6.52%	751s	5 bits
LogLog register	1,048,576	78.84%	1,690 s	5 bits
LogLog	4,522,232	8.76%	2,112 s	5 bits
HyperLogLog	4,983,171	-0.54%	2,907 s	40 KB
KMinValues	4,912,818	0.88%	3,503 s	256 KB
ScalingBloom	4,949,358	0.14%	10,392 s	11,509 KB
Datrie	4,505,514[d]	0.00%	14,620 s	114,068 KB
True value	4,956,262	0.00%	-----	49,558 KB[e]

[a] Processing time has been adjusted to remove the time to read the dataset from disk. We also use the simple implementations provided earlier for testing.

[b] Structure size is theoretical given the amount of data since the implementations used were not optimized.

[c] Since the Morris counter doesn't deduplicate input, the size and relative error are given with regard to the total number of values.

[d] Because of some encoding problems, the datrie could not load all the keys.

[e] The dataset is 49,558 KB considering only unique tokens, or 8.742 GB with all tokens.

Image 10

Running the simplest approach and the hyperloglog function we can see how much accurate is the algorithm:

```
In [115]: runfile('C:/Users/teo/untitled7.py', wdir='C:/Users/teo')
4553
4571
```

As we said before hyperloglog uses a spherical approach which means that splits the whole stream into buckets and count the longest observable run of zeroes an item can have. This is called stochastic averaging and can be really helpful in estimating unique items. We have to mention that the run of zeroes occurs from the hashed value of the input, that means that like count min sketch hyperloglog uses the similar hash function idea approach. Below we can see the math behind the algorithm, its simple a series of multisets multiplied with a hash variable for every bin.

$$E := \alpha_m \cdot m^2 \cdot \left(\sum_{j=1}^{m} 2^{-M[j]} \right)$$

where

$$\alpha_m := \left(m \int_0^\infty \left(\log_2 \left(\frac{2+u}{1+u} \right) \right)^m du \right)^{-1}$$

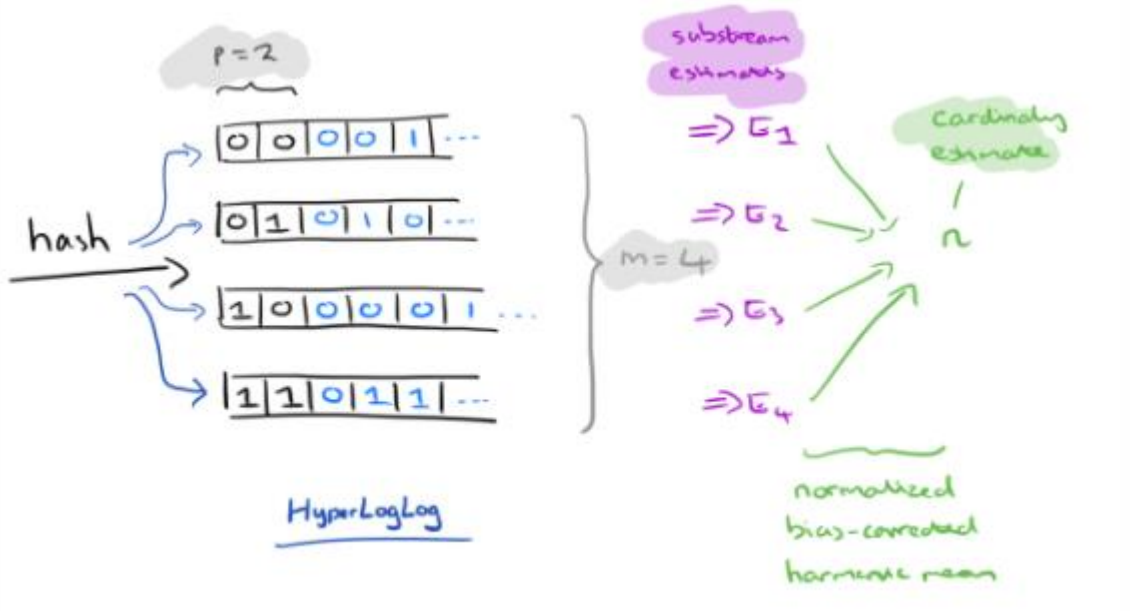
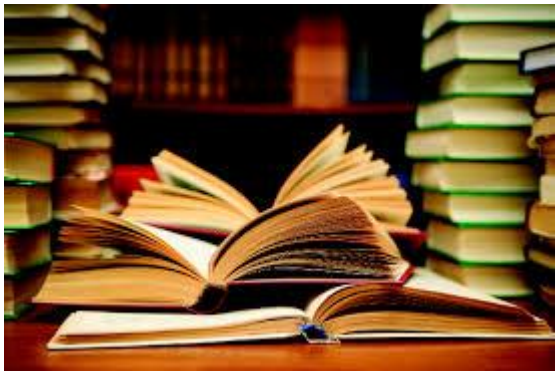


Image 11



Bibliography

[1] Rokach, Lior; Maimon, O. Reilly(2014). High Performance Python
[2] Flajolet, Philippe; Martin, G. Nigel (1985). "Probabilistic counting algorithms for data base applications"
[3] Dunham Margaret (2003), Data Mining: Introductory and Advanced Topics. Prentice Hall
[4] Han, J., Kamber, M., Pei, J. (2011), Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science
[5] Graham Cormode, and S. Muthukrishnan. LATIN 2004: Theoretical Informatics (2004) [6] Mitchell Tom (1997): "Machine Learning", McGraw-Hill.
[6] Pyle D. (1999): "Data Preparation for Data Mining", Morgan Kaufmann Publishers.
[7] Roiger J. Richard and Geatz W. Michael (2003): "Data Mining: A tutorial-based primer", Addison - Wesley
[8] Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop (2010)
[9] Argesti,A.(2002).Categorical Data Analysis 2nd edition,.Wiley