Practical Lab with Storm, 2nd part

THIS PAGE IS AVAILABLE AT : https://goo.gl/4ffvdC

A. COURSE MATERIAL

Hereafter, you can find all the lecture slides of this seminar :

- Lecture 1: Big Data Panorama - https://goo.gl/Bo1qX1
- Lecture 2: Data stream analysis, model and background - https://goo.gl/ehLAeJ
- Lecture 3: Similarity Metrics of distributed data stream - https://goo.gl/RyMA5O
- Lecture 4: Distributed monitoring: the need of sampling - https://goo.gl/M9xLva
- Lecture 5: Enhancement of a core sketching algorithm - https://goo.gl/gcG1HC
- Lecture 6: usage of stream analysis methods for stream processing - https://goo.gl/Vd0hWY

Email : Yann.Busnel@ensai.fr

B. PRACTICAL LAB WITH STORM, 1st PART

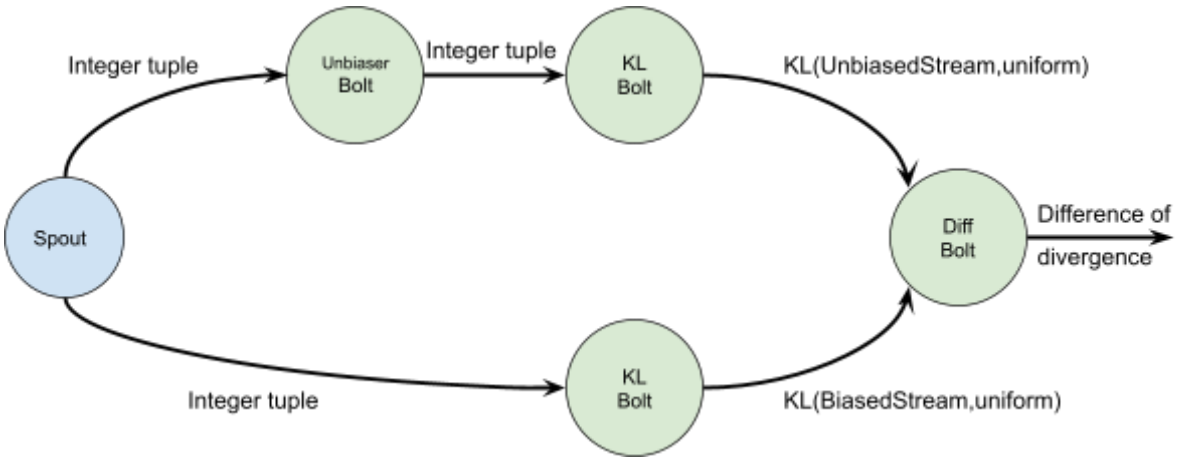
Before going any further, you must have finished the first part of the lab : https://goo.gl/Pz2uPT

C. UNBIASING A STREAM AND COMPARING IT WITH THE ORIGINAL ONE

In this lab, you will implement an advance topology in storm, with two main objective :

- Use of the technique provided in the following paper to unbiased a stream with a skewed distribution :
  Emmanuelle Anceaume, Yann Busnel, Bruno Sericola. Uniform Node Sampling Service Robust against Collusions of Malicious Nodes. Dans the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2013), Budapest, Hungary, June 2013.
- Compare the initial stream distribution and the unbiased one to estimate the gain of the aforementioned approach. This comparison should use the Kullback-Liebler divergence with the uniform distribution, for both streams, and send the difference of these values.

Below, you can find the topology abstraction to be implemented:



The initial **Spout** will generate stream of Random Integers, following a given biased distribution. In order to implement it, you can use the famous Commons Math library, from Apache : https://commons.apache.org/proper/commons-math/
It can provide to you a collection of classical distribution as Uniform, Zipfian, Poisson, etc. : https://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/distribution/package-summary.html

The **Unbiaser Bolt** will receive this stream and execute the sampling algorithm provided in the aforementioned paper (cf. Lecture 4). You then have to implement the Count-Min sketch (cf. Lecture 2) and the sampling memory in this bolt. At each reception, the bolt update the data structure (CM and Sampling memory) and emit an integer extract uniformly at random from the latter.

The **KL Bolt** has to compute the Kullback-Leibler divergence from the input stream and an uniform one:
https://en.wikipedia.org/wiki/Kullback–Leibler_divergence
Recall that the KL-divergence is provided by (cf. Lecture 3):

$$\mathcal{D}(p||q) = \sum_{u \in N} p_u \log \frac{p_u}{q_u} = H(p,q) - H(p),$$

where *p* and *q* are two probability distributions.
We can then rewrite the last equation with the empirical distribution.

$$\mathcal{D}(q_\sigma||p^{(\mathcal{U})}) = \sum_{i=1}^{n} q_i \log(q_i) - \sum_{i=1}^{n} q_i \log\left(p_i^{(\mathcal{U})}\right)$$

$$= \log(n) - \log(m) + \frac{1}{m}\sum_{i=1}^{n} m_i \log(m_i)\,.$$

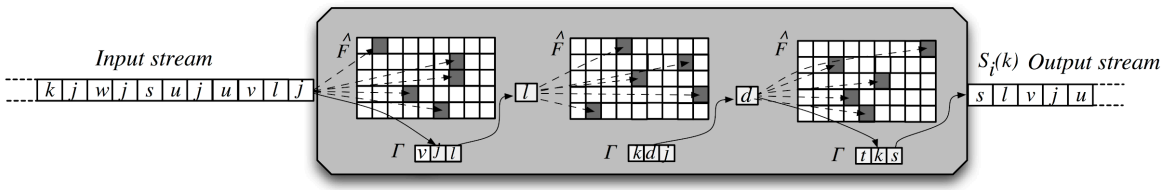
You will then need to track in this bolt the exact frequency vector of the stream, and the number of tuples received so far, and the number of distinct item.
Upon reception of a new tuple, you will have to update the data structure and to emit the updated value of the KL-divergence.

Finally, the **Diff Bolt** will make the difference between the last values received by the two instance of KL Bolt (one from the biased one, representing the baseline and one from the unbiased one), normalized by the biased one, that reflect the gain of the sampler.
You will have to differentiate the input stream (think on getSourceStreamId() in the Storm interface Tuple for instance).
Hint: to synchronize the diff value, you should also emit, with the KL-divergence value, the current *m*, as a time stamp. You then will also have to implement a dynamic buffer in case of desynchronization of input streams.

Finally, you should store these values in a log file, which will be plotted after execution to see the evolution of this gain according to time.

D. IF YOU HAVE TIME

Exploit the above result by showing that processing the input stream with sketches put in series decreases this convergence time, and this is achieved without requiring any additional space nor additional operations per item.
For instance, consider the following cascade:



This can be easily obtained with storm, as you just have to put several Unbiaser Bolt in series (take care to reduced consequently the memory usage of each Bolt to remain fair with respect to the previous topology memory usage).

Compare the evolution of the gain with the previous one.