ACM Reproducibility Step-by-Step Guide

To Replicate First Result

1. System configuration to reproduce our result in the paper: Section 4.1, first paragraph of https://arxiv.org/pdf/1709.01190.pdf
2. git clone https://github.com/RUSH-LAB/Flash.git
3. Navigate to the FLASH folder
4. Download the Webspam dataset from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/webspam_wc_normalized_trigram.svm.bz2
5. Download the Webspam groundtruth from https://github.com/wangyiqiu/webspam
6. Place the dataset and the groundtruth files in a directory you like
7. Go to link https://github.com/RUSH-LAB/Flash, search for keyword "Tutorial". Follow the instructions in the Tutorial section step by step exactly
8. After finishing the Tutorial section, on the same page, search for keyword "System Configuration", edit LSHReservoir_config.h and set the configuration to be "CPU Only" for sparse dataset
9. Type make clean; make
10. ./runme to run the experiment
11. From this you would be able to see the quality result and timing result for the webspam dataset

Interpret meaning of output script

Now the first set of result is replicated successfully, here is a breakdown of how to interpret the output that you have sent back:

```
Reading groundtruth ...
Completed.
Reading data ...
[readSparse]
[readSparse] Read 1304697446 numbers, 350000 vectors.
Completed, used 452806ms.
Adding data to hashtable / Preprocessing / Indexing ...
Batch 0(datapoint 10000), already taken 146 ms.
Table Mem Usage ranges from 0.037659 to 0.075653, average 0.059171
Batch 5(datapoint 44000), already taken 439 ms.
Table Mem Usage ranges from 0.116302 to 0.258575, average 0.190035
```

```
Batch 10(datapoint 78000), already taken 672 ms.
Table Mem Usage ranges from 0.167419 to 0.373444, average 0.270504
Batch 15(datapoint 112000), already taken 869 ms.
Table Mem Usage ranges from 0.207153 to 0.457947, average 0.331152
Batch 20(datapoint 146000), already taken 1066 ms.
Table Mem Usage ranges from 0.242065 to 0.523254, average 0.380833
Batch 25(datapoint 180000), already taken 1249 ms.
Table Mem Usage ranges from 0.269989 to 0.575989, average 0.422205
Batch 30(datapoint 214000), already taken 1415 ms.
Table Mem Usage ranges from 0.295258 to 0.617493, average 0.457644
Batch 35(datapoint 248000), already taken 1587 ms.
Table Mem Usage ranges from 0.319458 to 0.656647, average 0.489912
Batch 40(datapoint 282000), already taken 1774 ms.
Table Mem Usage ranges from 0.341827 to 0.688721, average 0.518115
Batch 45(datapoint 316000), already taken 1952 ms.
Table Mem Usage ranges from 0.362427 to 0.717407, average 0.543713
Completed, used 2068ms (indexing time).
Querying...
Queried 10000 datapoints, used 311ms (querying time).
[similarityMetric] Averaging gtruth.
[similarityMetric] Averaging output.
```

The rest of the result script are the accuracies using different criteria, for example:
S@k (criteria name) = s_out(s_true): **In top k, average output similarity (average groundtruth similarity). (criteria meaning)**

```
S@1 = 0.915 (0.972)
S@10 = 0.876 (0.957)
S@20 = 0.857 (0.950)
S@30 = 0.843 (0.946)
S@32 = 0.841 (0.945)
S@40 = 0.830 (0.942)
S@50 = 0.819 (0.939)
S@64 = 0.806 (0.936)
S@100 = 0.781 (0.930)
S@128 = 0.766 (0.926)
```

(criteria readings, above)
(output in vector format, easier to copy, paste and plot, below)

```
1 10 20 30 32 40 50 64 100 128
0.915 0.876 0.857 0.843 0.841 0.830 0.819 0.806 0.781 0.766
0.972 0.957 0.950 0.946 0.945 0.942 0.939 0.936 0.930 0.926
```

…. (the rest of the criteria has the same format)

Example of how to replicate Figure 7 of paper

Let’s focus on the two diagrams on the **right** of figure 7, they use the same dataset webspam as you are using right now:

- The vertical axis is **R@100**, a kind of criterion, which is available from the result script directly
- The horizontal axis are **construction and query time,** which are available from the result script directly (see section above)

To make these two number change such that they form a plot like shown on the diagram, vary parameter NUMTABLES (corresponding to param L in the paper) in benchmarking.h (the default value is 32, try 16, 32, 64, 128 … etc). Every time the parameter is changed, one needs to **recompile and rerun the program** to get the **new reading**. In other words, every “run” correspond to one data point on the plot. Similarly, to reproduce figure 4,5,6, vary parameter L,K,R in benchmarking.h. Below is a table summarizing the correspondence between parameter names used in the paper and those used in benchmarking.h

Paper	benchmarking.h
L	`NUMTABLES`
K	`K`
R	`RESERVOIR_SIZE`

Part of the paper also mentions the index size, this is observed directly from the terminal by tying “top” and observe the maximum memory usage.

To replicate result of other dataset

Datasets are here: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html
We used:

- Url
- Webspam
- Kdd12

Groundtruths are here (corrected):
~~https://drive.google.com/file/d/1G3zMwrutuyf3zc4XWZAYQH2SczVHq1H5/view?usp=sharing~~
https://drive.google.com/drive/folders/1WUPmrMFay4k2Au_0F3n-5GLgUCrxAS0a?usp=sharing
The testing process is identical to that of webspam that you have successfully replicated. The default parameters are also in benchmarking.h, just remember to toggle the correct parameter section by commenting/uncommenting the macro on the top of benchmarking.h.

To change CPU/GPU configuration

Parts of the paper has results for CPU and parts have result for GPU
I would recommend that you start with the CPU results (which you have successfully replicated), then to change system configuration, see System Configuration section of https://github.com/RUSH-LAB/Flash
Basically, for GPU results, uncomment the following two lines on the top of LSHReservoirSamplier_config.h:

```
//#define OPENCL_HASHTABLE // Placing the hashtable in the OpenCL device.
//#define OPENCL_HASHING   // Perform hashing in the OpenCL device.
```