How to organise expts for ass2

avo@unimelb.edu.au

for use in comp20003

Experiments

Sample size:

```
n=  1000  2000  4000  8000  16000 ?
n=   3000 6000 9000 12000 15000 18000 ?
```

Data type: sorted, random ?

query: run 100 queries each times and compute average?

Stage 1 vs stage 2?

We want to have the following table/file and use it to build graphs:

stage	data_type	n	comparison/query
1	sortx	1000	???
1	sortx	2000	???
…			
1	rand	1000	???
..			
2	sortx	1000	???
2	sortx	2000	???
…			

Creating data file

1. Create a sorted data file with no header line

```
tail -n +2 CLUEdata2018_sortx.csv  > sortx_all.csv
```

2. Create 2 data files of 1000 lines: run fhe following 2 commands:

```
head -n 1000 sortx_all.csv > sortx_1000.csv
shuf sortx_1000.csv >  rand_1000.csv
```

Repeat step 2 for all the sizes you want.

Or, you can also run the following single-line command (note: you need to join the lines together to make a single line):

```
for size in 1000 10000 100000;
do head -n $size sortx_all.csv > sortx_$size.csv;
shuf sortx_$size.csv >  rand_$size.csv; done
```

Note: The above 3 parts are in a single line, if you want to break into sub-lines, add a single slash \ to the end of each sub-line.

- The data files don't have the header line. It's OK for the experiment purpose.

Creating query files

Create 2 query files of 100 lines each (q1.txt and q2.txt for Stage 1 and Stage 2 from sortx_all.csv : Run *two* **single-line** commands

```
cat sortx_all.csv
    | awk -F ',' '{if(($9~/^[0-9]+/) && ($10~/[0-9,.]+/) ) {print $9,$10;}}' |
shuf  > ./x
```

```
head -n 100 ./x
    | awk '{printf("%lf %lf\n", $1, $2)}'
    > q1.txt
```

Note: You should check the content of `q1.txt` after that to make sure you've got a non-empty and well-formatted query file.

For Stage 2 queries, change the last 2 components to

```
    | awk '{printf("%lf %lf 0.0005\n", $1, $2)}'
        > q2.txt
```

Note: **each command is in a single line**, if you want to break into sub-lines, add a single slash \ to the end of each sub-line.

Summing up the output of you program

Suppose that the output into stdout of Stage 1 are *only* of format:

```
144.959522 -37.800095 --> 4000
0 0 --> 300
```

And we run, say:

```
./dict1 sortx-1000 o.txt > S1out_sortx_1000.txt
```

Then (**$4** for "column 4", which is the number 4000 and 300 in the top 2 lines):

```
cat S1out_sortx_1000.txt | awk 'BEGIN{sum=0; n=0}{sum = sum+$4;
n++}END{print "1000 " sum/n}'     >> Stage1_sortx.txt
```

will *append* line

```
1000 2150
```

to file `Stage1_sortx.txt`. This line represents:

```
n    average_cmp_per_search
```

Note:

Make sure that your output to stdout is in correct format: `0<SPACE>0<SPACE>--><SPACE>300`

For Stage 2 output, we need to relace **$4** by **$5** because the output has format where num_cmp is in column 5:

```
144.959522 -37.800095 0.0005 --> 4000
```

SUMMARY: Process for expts with sizes 100 200 300

Creating a sorted all-record data file:

```
tail -n +2 CLUEdata2018_sortx.csv  > sortx_all.csv
```

Creating data files for experiments with n= **100 200 300** (you should change these to your designed values such as : **3000 6000 9000 12000 15000 18000**

```
for size in 100 200 300; do head -n $size sortx_all.csv >
sortx-$size.csv; shuf sortx-$size.csv >  rand-$size.csv; done
```

Run experiments (warnings: it might take long, depending on how fast is your code):

```
for map in 1 2; do for size in 100 200 300; do for type in
sortx rand; do ./map$map $type-$size.csv out-$map-$type-
$size.txt < q$map.txt > map$map-$type-$size.txt; done ; done;
done
```

Summary all expt outcome into file **expts.txt**

```
for map in 1 2; do for type in sortx rand; do for size in 100
200 300; do cat map$map-$type-$size.txt | awk -v size=$size -
v map=$map -v type=$type 'BEGIN{sum=0; n=0}{sum = sum+$4;
n++}END{print "map"map, type, size, sum/n}' >> expts.txt ;
done ; done; done
```