We thank all the reviewers for their constructive feedback and comments. We will include the new experiments in full scale for the final version of the paper, and will incorporate all comments and suggestions on the clarity of presentation.

To reviewer #1: Our experiments were focused on varying the number and size of the buckets (by changing the parameters k and L). Per your suggestion, we ran our experiments for two other datasets: SIFT10K and GloVe (random subset of 10k words out of 1.2M words). These along with MNIST are standard benchmarks in the context of Nearest Neighbor algorithms (see [ABF17]). We use the same tuning method for both datasets and further vary k and L to get different size buckets. The end results are slightly better than the results on the MNIST. For example, for the same tuned parameter ($k = 15$ and $L = 100$), we get the following. For SIFT, our algorithm performs 1.35 times worse than the optimal algorithm (with the same measure of statistical distance to the uniform distribution), while the other two approaches perform 6.04 times worse and 9.62 times worse than the optimal. For GloVe, our algorithm performs 2.42 times worse but the other two perform 5.94 and 11.84 times worse respectively.

We used the first 10K vectors in MNIST in order to reduce the amount of randomization. The number of occurances of the digits from 0 to 9 are $[1001, 1127, 991, 1032, 980, 863, 1014, 1070, 944, 978]$. We further ran the experiments again with the random selection of the entries and here are the results for the base case of $k = 15$ and $L = 100$: Our approach performs 2.42 times worse, while the other two approaches perform 6.52 times and 9.87 times worse than the optimal.

We will revise the introduction part of the paper to make it more succint and postpone the discussion about importance and other applications of sampling from sub-collection of sets to the appendix.

In order to have a meaningful comparison between distributions, in our code, we retrieve a random neighbor of each query $100m$ times, where m is the size of its neighborhood (which itself can be as large as 1000). We further repeat each experiment 10 times. Thus, every query might be asked upto 10^6 times. This is going to be costly for the optimal algorithm that computes the degree exactly. Thus, we use the fact that we are asking the same query many times and preprocess the exact degrees for the optimal solution. Therefore, it was not meaningful to compare runtimes directly. We are running the experiments on a smaller size dataset just to show the difference in the runtimes of all the four approaches: Our sampling approach is twice faster than the optimal algorithm, and almost five times slower than the other two approaches. However, when the number of buckets (L) increases from 100 to 300, our algorithm is 4.3 times faster than the optimal algorithm, and almost 15 times slower than the other two approaches.

As you suggest, we can show a trade-off between our proposed sampling approach and the optimal. For example, by asking twice more queries (for degree approximation), the solution of our approach improves from 2.5 to 1.58, and with three times more, it improves to 1.21, and with four times more, it improves to 1.05. We will include the full graph.

To reviewer #4: The main criteria of our data structure is that even if you ask the same query multiple times, you get a point which is independently almost uniform each time. In order to achieve this using L_0 sampler, we need to have an L_0 sampler per query. We dont consider a linear dependence on the number of queries to be desired as one can get a much simpler data structure (simple variant of LSH) for a linear dependence. We remark that in the L_0 sampler setting, the algorithm needs to work in a more restricted model.

For tuning the parameters of LSH, we follow the method described in [DIIM04] (see the references of the main submission file), and manual of E2LSH library [And05]. Nevertheless, we further vary k and L to get buckets of larger size, more outliers, and larger number of buckets.

Certainly one can get a tradeoff by approximating the degree more precisely. Please see above (lines 27-29).

Our sampling approach provides an algorithm with dependency on the number of points (n) which matches that of the optimal LSH, while providing an almost uniform distribution. However, it is an interesting open question to find the tight dependency on the density of the neighborhood of the query ($|N(q, cr)|/|N(q, r)|$).

In general, to meaningfully compare the distributions, we might ask for a random neighbor of each query 10^6 times (please see lines 18-20 above for details). This is why we did not afford to run it on huge datasets. However, we expect that the algorithm performs similarly on larger datasets. Per your suggestion, we ran our experiments for slightly larger dataset (100k word representaion from the GloVe dataset that has smaller dimension), and the results are as follows. For the base case ($k = 15$ and $L = 100$), our approach performs 1.41 times worse than the optimal algorithm but the other two approaches perform 3.73 and 6.01 times worse.

References

[ABF17] M. Aumüller, E. Bernhardsson, and A. Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In *International Conference on Similarity Search and Applications*, 2017.

[And05] Alexandr Andoni. E2lsh 0.1 user manual. *https://www.mit.edu/ andoni/LSH/manual.pdf*, 2005.