Review 4

Bilal Akliai

4 Jul 2024

What I did I read the paper about the NN-Descent algorithm. I implemented it in Python, but it was not really efficient (I didn't try to optimize it because parallelization seemed to be not really efficient in Python).

So, I switched to C++. I implemented all the algorithms that I had before + the NN-Descent algorithm. It uses Python to construct the fingerprints, and then with Pybind11 it works in C++.

I added parallelization in the NN-Descent algorithm. Thanks to it, we can see a real difference of insertion time.

For example, for 30k compounds, it takes only 2.8s for the insertion time. For the search time, it takes 4s without HNSW, and 6.6ms with HNSW (similar to what we had in Python 2 weeks ago). The problem is that we have only a score of 28%. I think that it can be an error of implementation (I noticed an error in my implementation of the NN-Descent algorithm and I didn't have time to change it for today). So, I will try to correct it, and I hope that it will improve the result.

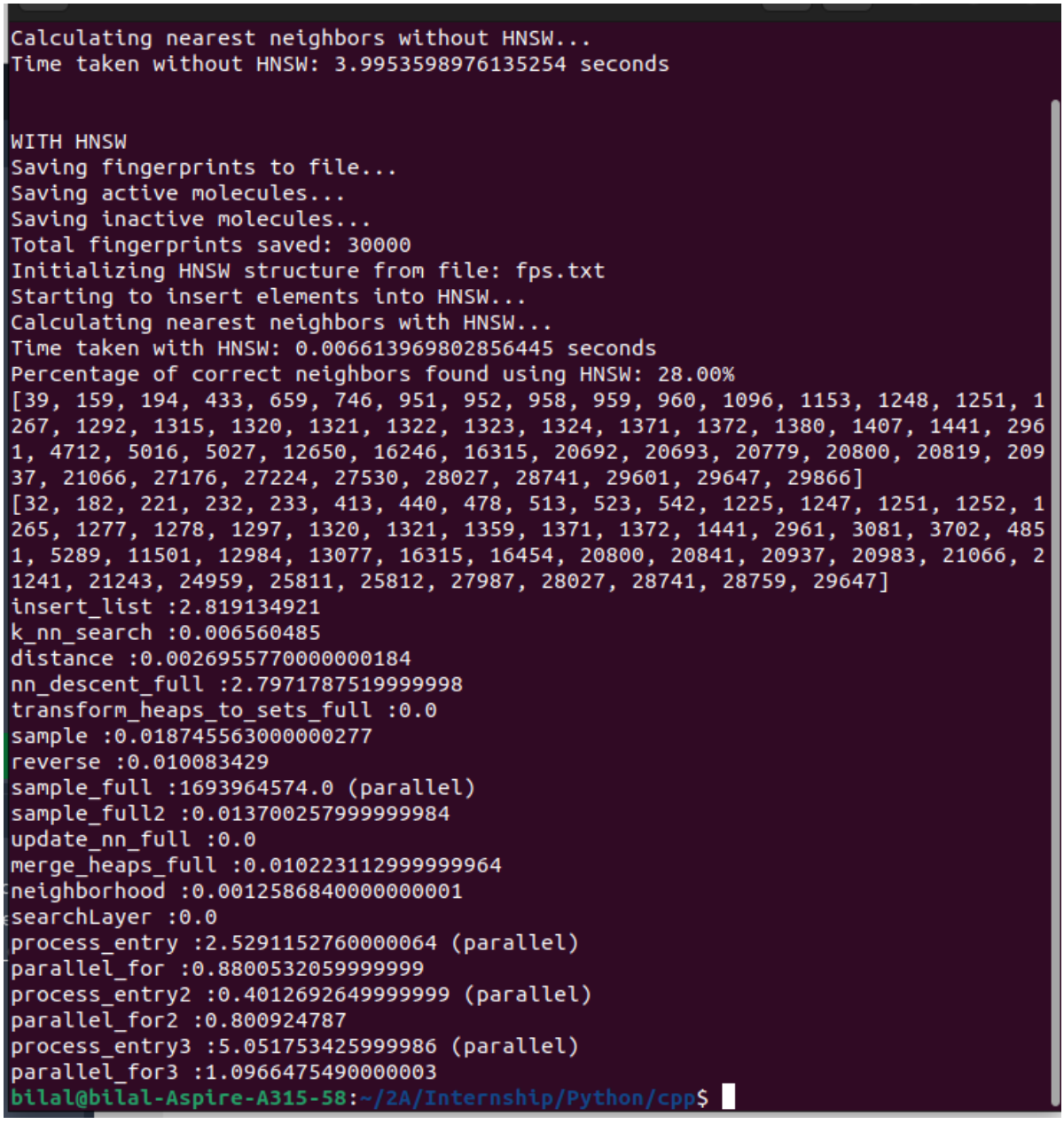

Figure 1: 50-KNN for 30k compounds, with K=50, efConstruction=100, rho=5 and delta = 0.001

For 110k compounds, it takes 11.3s for the insertion time (compared to about 6min in Python 2 wekks ago). So, we really improved that time! For the search time it's also correct (about 8.2ms). However, we still have the same problem: the result is not good !!

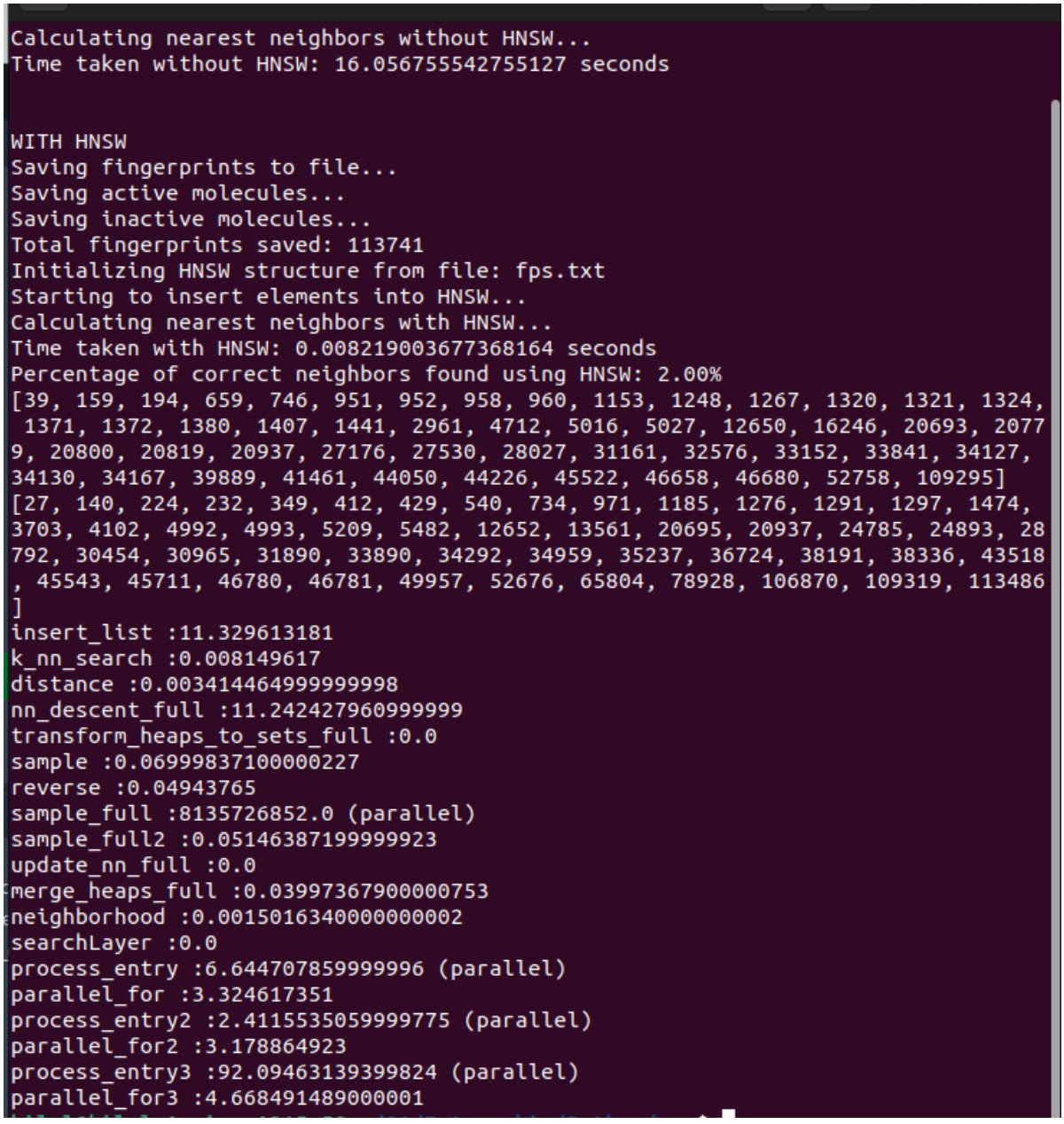

Figure 2: 50-KNN for 110k compounds, with K=50, efConstruction=100, rho=5 and delta = 0.001

By changing parameters, I can have good results. For example, if I put K=400 (number of neighbors for each node) and efConstruction=800. I can have 90% which is very good. However, the problem is that changing these parameters impact a lot the memory (400 neighbor for each node is too much!!) and these 2 parameters increase also a lot the search time: so it is not what we want. Indeed, the search time is now about 200ms (compared to about maximum 10 with normal parameters). The insertion time is still very good (22s compared to 6min in Python).

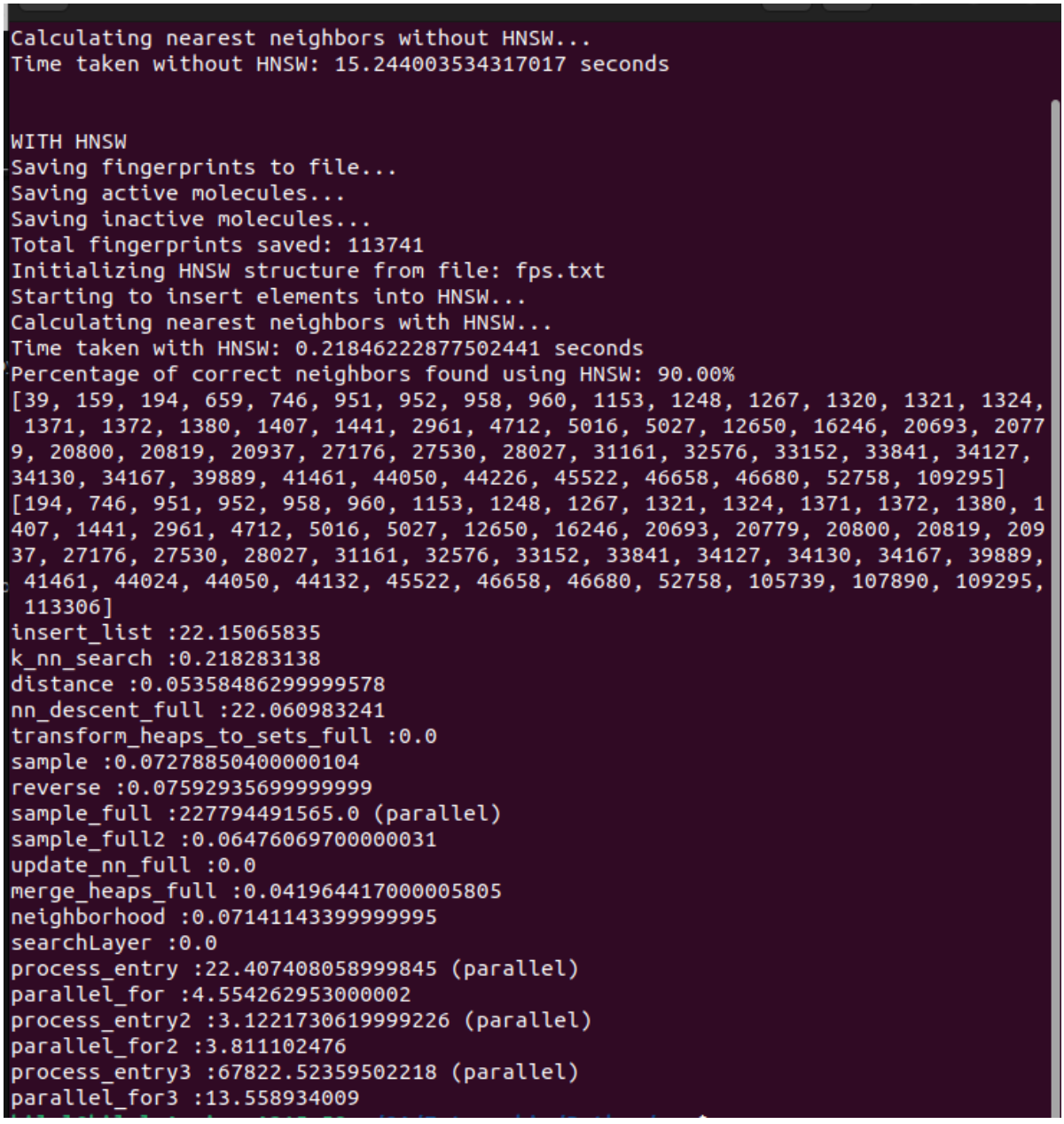

Figure 3: 50-KNN for 110k compounds, with K=400, efConstruction=800, rho=5 and delta = 0.001

So, the objective is to have this kind of result but with small parameters (so, it will be good for the memory, and the search time). I will try to correct my implementation of the NN-Descent algorithm (maybe it will solve this problem).

For 200k compunds we have also good results: 78% with an insertion time of 45s. However, it's also with big parameters, so the search time is very long (about 0.3s).

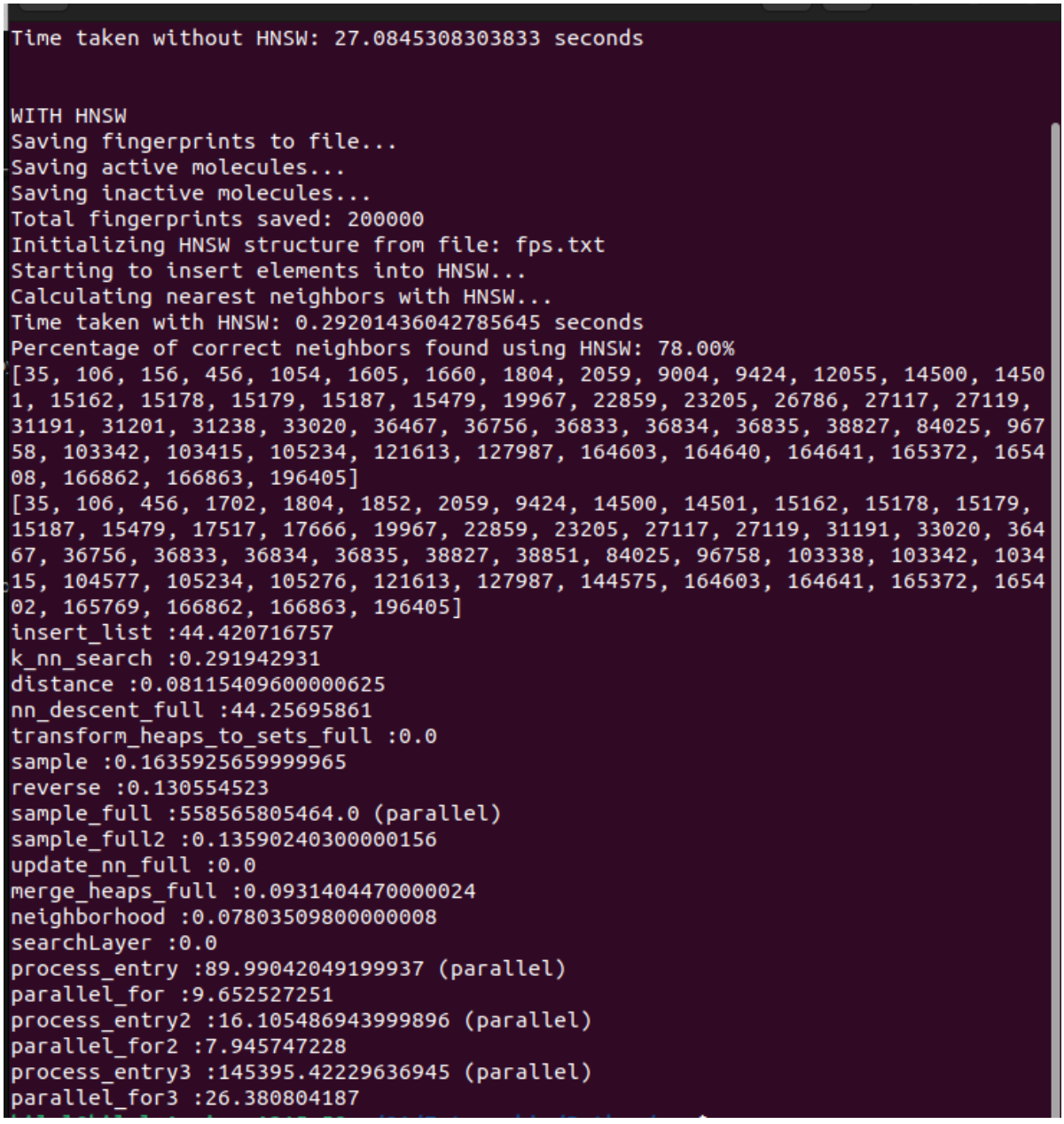

Figure 4: 50-KNN for 200k compounds, with K=400, efConstruction=800, rho=5 and delta = 0.001

I have also another problem. I have to optimize the memory, because it don't work for more than 200k or 300k compunds (the process is killed cause of not enough memory). I've tried not to store all the fingerprints but only the positions with "1 "s (as there are a lot of "0 "s) but this doesn't allow me to process many more molecules.

What I still have to do For the moment, I didn't try to optimize the search time (use techniques for Jaccard similarity for accelerating computations like we said for example...).

Objectives I think that firstly I have to resolve the problem of accuracy (without using big parameters to maintain a good search time). Then, I have to optimize the memory to be able to test with more compounds. Next, I will have to try to optimize the search time (it seemed to be not really adapted to parallelization: I tried it a bit but I didn't succeed). And, finally I will have to use the latest papers about HNSW to have the more efficient structure as possible.