Efficient Processing of k Nearest Neighbor Joins using MapReduce

I. Summary

The paper illustrates a efficient processing of kNN Joins using MapReduce

In the preliminaries, the authors shows the basic concepts of kNN Join, MapReduce Framework, and particularly the Voronoi Diagram-based Partitioning, which is the mathematic fundamental for partitioning the input set R and S.

A native and straightforward idea of performing kNN join in MapReduce is similar to the hash join algorithm. R is split into disjoint subsets, each subset R_i is distributed to a reducer, and without any pruning rule, the entire set S has to be sent to each reducer to be joined with R_i.

In order to reduce the shuffling cost, a better strategy is that R is partitioned into N disjoint subsets and for each subset R_i, find a subset of S_i that $R_i \ltimes S = R_i \ltimes S_i$ and $R \ltimes S = \bigcup_{1 \leq i \leq N} R_i \ltimes S_j$. So S_i is sent to the reducer that R_i belongs to and the KNN join is performed between R_i and S_i only.

So given the input set R and S, the partition of R_i and S_i should be calculated first in the map procedure, and then each R_i,S_i pair will be sent to the same reducer to execute KNN join respectively. The details are shown below, and it takes three steps to complete the kNN Join.

- First, the master node invokes a preprocessing step, which takes the original R and S as input and finds out a set of pivot objects based on the input dataset R. The pivots can be got by Random Selection, Farthest Selection or k-means selection, and the pivots is used to create a Voronoi diagram, which can help partition objects in R effectively while preserving their proximity.
- Second, the first MapReduce job consists of a single Map phase, which takes the selected pivots and datasets R and S as the input. The output of the mapping phase is a partitioning on R, based on the Voronoi diagram of the pivots. Meanwhile, the mappers also collects some statistics about each partition R_i.
  The partitioning on R can be represented as a table, which including each object o along with its partition id, original dataset name (R or S), distance to the closest pivot.
  The tatistics are kept in two summary table T_R and T_S. T_R maintains the following information for every partition of R: the partition id, the number of objects in the partition, the minimum distance L(P_i^R) and maximum distance U(P_i^R) from an object in partition P_i^R to the pivot.