Sampling for Join Discovery

C Rajmohan, Vamshi Pasunuru
ME, CSA, IISc Bangalore

1 Problem Statement

Given two tables, the join discovery problem is to find out all possible ways in which the two tables can be joined to produce significant results. Each way corresponds to a mapping of attributes from one table to the other. The strength of a join for a mapping M from table T1 to T2 is the percentage of rows from T1 that join to some row of T2 under the mapping M. We are interested in finding all the mappings that have support greater than a threshold t. This is a very expensive operation since the number of possible mappings is exponential in the number of attributes. Thus, we want to consider sampling techniques for join discovery.

2 Proposed Technique

Lets say we have two data sets namely Table A(a1,a2,....,am) and Table B(b1,b2,....,bn) and we want to find potential join attribute pairs between A and B.

Given relation A and B in flat files, our algorithm works in 3 phases.

2.1 Phase 1

- **Determine datatypes** of each attribute in A and B. This requires one linear scan of the dataset(sampling can be tried out).
- Eliminate non-compatible attributes by datatypes.

Datatype Determination Technique Say we want to determine datatype of A.a1 attribute. Candidate datatypes are Integer[1], Numeric[2], String[3] and Date[3]. The numbers in bracket denote rank of the datatype. Ranking is defined to give a priority for data type of an attribute.

Initially rank of an attribute will be zero. Then we start scanning the tuples values of that attribute. After each scan of a tuple value for a1, rank is updated as follows.

$$rank(a1) = max(rank(a1), rank(CurrentType(a1))$$

We can stop scanning, once rank(a1) reaches 3 without waiting for the complete scan of the dataset.

2.2 Phase 2

Out of the remaining combinations, we will use the following techniques to prune some more combinations by taking advantage of specific datatypes.

Integer/Numeric Collect the following statistics for both relations by a linear scan(sampling can also be used).

- Mean
- Squared sum
- Linear sum

- Variance
- Unique or not
- Increasing or Decreasing etc.

Compare statistics of relations A and B. If a pair is not a join pair, it is likely that above parameter values differ significantly. For example, mean value differs by a huge margin when age attribute of table A is compared with zipcode attribute of table B. How much statistics should differ for eliminating a combination needs to be studied further. For instance, we can assign a weight to each of the above parameter and find the weighted sum W. If W is above some threshold, it is a potential join pair. Otherwise we can prune it with high confidence. This weighted sum can also be used along with join support to order the results at later stage.

String Since statistics are not much helpful here, we have come up with the following idea to efficiently prune some of the candidate pairs.

$Sample-JOIN()$ is the traditional join between Table A's a1 attribute and Table B's b1 attribute on a sample of size s. It returns number of rows from Table A which had a match with some row of Table B.

Parameters to be configured for the algorithm are

- Threshold t (in percentage). It is the minimum number of rows in A, that join with some row of B.
- Sample Size s (in percentage)
- Sampling Repeat Count c. Number of times to repeat sampling
- Relaxation parameter α. It is in (0, 1]. 1/2 is suggested.

Take a sample of size s from relations A and B and do a join. Find support. If support is below scaled threshold $st(st = s * t)$, repeat. Sampling can be repeated upto c times. Compute sum of support. If it is less than α times ideal sum of support$(t * s * size(A) * c)$ prune it.

Heuristics

- **Categorical attributes** Database metadata does not tell us if its string or categorical. we need to write a helper for that. For now lets assume there's one. Since the number of categories are limited, so its not unique which implies that sampling can work well.
- We expect that $Sample - JOIN(a1, b1)$ will be close to scaled-threshold (which is $t * s$) , to avoid False Negatives we will run this for c times. So $SUM(Sample - JOIN(x, y)) >= (t * s * c * size(A) * alpha)$ should hold. Intuitively it specifies how strict the condition is, if its 1 the algorithm expects that in all runs of sampling we get at-least the scaled-threshold. The idea is to set it to something like 1/2 or 3/4 so that even if in some run the algorithm has less than scaled-threshold it will pass this phase.

2.3 Phase 3

We muse consolidate the findings of phase 1 and phase 2 and present potential join pairs in support percentage order.

3 Experiments

Hive and Hadoop or postgreSQL. Must have good support for sampling, partitioning.