Data Linkage for BigData using Hadoop MapReduce

[1]Bargavi C, [2]Blessa Binolin Pepsi M
[1,2]Dept. of IT, Mepco Schlenk Engineering College, and Sivakasi, Tamil Nadu, India

Abstract
One-to-many data linkage is important in datamining. In earlier works data linkage is performed among entities of the same type. To link between matching entities of different types in larger datasets a new one-to-many data linkage method with mapreduce is proposed that links between entities of different natures. The proposed method is based on a one-class clustering tree (OCCT) that characterizes the entities that should be linked together. The tree is built using hadoop mapreduce framework for linkage. Using mapreduce for linkage results in a reduced execution time due to its distributed environment. It also results in an efficient processing of larger datasets.

Keywords
Datamining, Hadoop, Mapreduce, Clustering Tree

I. Introduction

Data linkage is one of the important task in data mining.It is of two kinds: one-to -one and one-to-many. One-to-one data linkage associates one entity with a matching entity in another data set. One-to-many data linkage associates an entity with a group of matching entities from another data set. In this paper one-to-many data linkage is implemented with mapreduce framework.

Hadoop MapReduce is a software framework for distributed processing of large data sets. It is a sub-project of the Apache Hadoop project. The framework takes care of scheduling tasks, monitoring them and re-executing failed tasks. The primary objective of MapReduce is to split the input data set into independent chunks that are processed in a completely parallel manner. The Hadoop MapReduce framework sorts the outputs of the maps, which are then input to the reduce tasks. Both the input and the output of the job are stored in a file system.

The proposed method with mapreduce links between the entities using a One-Class Clustering Tree. A clustering tree is a tree contains a cluster in each of the leaves instead of a single classification. Each cluster is generalized by a set of rules that is stored in the appropriate leaf.One-class clustering tree is implemented with the help of mapreduce in parallel. Clustering tree assigns each linkage task to mapper class. Master node in the mapper assigns the work to slaves. Slaves inturn completes the work and returns the result to master node. The reducer class combines the intermediate results from mapper class to provide a complete clustering tree. As clustering tree is implemented in a parallel and distributed environment it reduces the execution time.

The objectives of the paper is threefold:
1. To implement one-to-many data linkage
2. To make data linkage for larger datasets more efficient
3. To execute data linkage in a parallel and distributed environment using mapreduce.

The rest of the paper is organized as follows: Section II reviews the related works on data linkage, section III provides an introduction to hadoop mapreduce programming environment, section IV briefs the data linkage problem, section V and VI provides the proposed solution, section VII concludes the paper.

II. Related Work

Data linkage is the task of matching entities from two different data sources. Some of the previous works on one-to-many data linkage has been addressed here. Storkey et al. [2] use the expectation maximization algorithm for combining two datasets.Ivie et al. [3] use one-to-many data linkage for genealogical research. Christen and Goiser [4] use a C4.5 decision tree to determine which records should be matched to one another. Maximum-likelihood estimation (MLE) [5] is used to determine the probability of a record pair being a match.One-class clustering tree [1] is used to implement linkage of two different record pairs.

In this paper, one-class clustering tree is implemented using hadoop mapreduce.The major advantages of using mapreduce includes following:
1. Parallel processing
2. Efficient implementation of large datasets
3. Reduced execution time.

III. MapReduce Framework

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master.

Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job and configuration to the JobTracker which then assumes the responsibility of distributing the software/ configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job.Input and Output types of a MapReduce job: (input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2> -> reduce -> <k3, v3> (output).

Map() function understands exactly where it should go to process the data. The source of data may be memory, or disk, or another node in the cluster. The program fetches data from a data source and is usually executed on the machine where the application is running. In MapReduce implementations, the computation happens on the distributed nodes.

Reduce() function operates on one or more lists of intermediate results by fetching each of them from memory, disk, or a network transfer and performing a function on each element of each list. The final result of the complete operation is performed by collating and interpreting the results from all processes running reduce() operations.

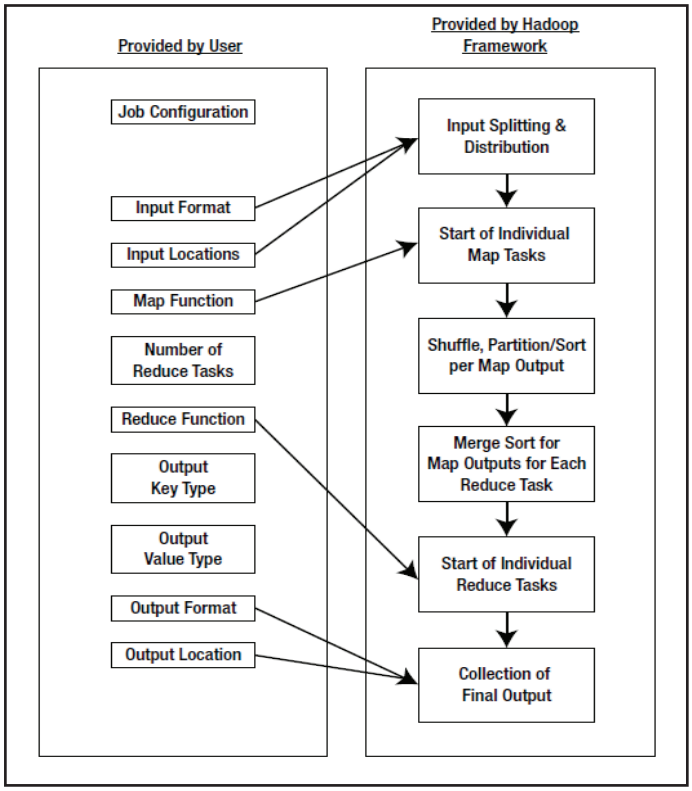

Fig. 1: Working of Hadoop Mapreduce Framework

Hadoop can run in three different ways, depending on how the processes are distributed:

A. Standalone Mode

This is the default mode provided with Hadoop. Everything is run as a single Java process.

B. Pseudo-distributed Mode

Here, Hadoop is configured to run on a single machine, with different Hadoop daemons run as different Java processes.

C. Fully Distributed or Cluster Mode

Typically, one machine in the cluster is designated as the NameNode and another machine as the JobTracker.

IV. Datalinkage Problem

Data linkage problem consists of two data tables that do not have a unique identifier. The objective is to match between the records of one table with their corresponding matching records in another table.

To solve this , one-class clustering tree is used. It matches records originating from one table with records from another table. The entities of one datatable may be of a different type of entities from another. The inner nodes of the tree represent attributes of one dataset. The leaves of the tree provide a compact representation of records from another dataset that needs to be linked on the path from the root to the leaf.

In one-class clustering tree each one-to-many record linkage task is done using mapper class. Reducer class combines the result of all the record linkage tasks.

V. Proposed Solution with Mapreduce

In proposed method, a linkage model is created for linking data sets. Linkage model is implemented with the hadoop mapreduce framework.

A. Linkage Model Creation

The clustering tree is induced using maximum likelihood splitting criteria. The splitting criterion is used to determine which attribute should be used to build the tree. Also, a pruning process is applied to decide which branches should be trimmed.

B. Maximum Likelihood Splitting Criteria

This splitting criterion uses the MLE(maximum likelihood estimation) to choose the attribute that is most appropriate to serve as the next splitting attribute. Each candidate attribute from the set of attributes splits the node data set into subsets according to its possible values. Goal is to choose the split that achieves the maximal likelihood, and the attribute with the highest likelihood score is chosen as the next splitting attribute in the tree.

C. Pruning

Pruning is an important part of the tree induction process. A good pruning process will result in a tree which is accurate and avoids overfitting. There are two common methods for pruning a decision tree: prepruning and postpruning. In prepruning, a branch is pruned during the induction process if none of the possible splits are found to be more beneficial than the current node. In postpruning, the tree is grown completely, followed by a bottom-up process to determine which branches are not beneficial.

Maximum-likelihood method is used for pruning. MLE score is computed for each of the possible splits. If none of the candidate attributes achieve an MLE score which is higher than the current node's MLE score, the branch is pruned and the current node becomes a leaf.

D. Leaf Representation

Once the model is built, each leaf has a data set containing the matching records from another dataset. Each leaf is represented by a set of probabilistic models for following reasons: I) To create a compact representation of the linkage model and ii)to be more generalized. These models represent the probability of an attribute, given the values if all other attributes.

E. Datasets Used

The OCCT was evaluated using data sets from three different domains: i)data leakage prevention, ii)recommender systems, and iii)fraud detection. In the data leakage prevention domain, the goal is to detect abnormal access to database records that might indicate a potential data leakage or data misuse. The goal is to match an action, performed by a user within a specific context, with records that can be legitimately retrieved within that context.

In the recommender systems domain, the proposed method is used for matching new users of the system with the items that they are expected to like based on their demographic attributes. In the fraud detection domain, the goal is to identify online purchase transactions that are executed by a fraudulent user and not the legitimate user.

F. Linkage with MapReduce

This section provides an overview of how linkage is performed with the Map/Reduce framework. The interfaces in hadoop will implement the linkage as follows:

1. Mapper

The dataset is given to mapper as input. Mapper maps input key/value pairs to a set of intermediate key/value pairs. Maps are the individual tasks that transform input records into intermediate

records. The number of maps is usually driven by the total size of the inputs, that is, the total number of blocks of the input files.

2. Reducer
Reducer reduces a set of intermediate matching records which belongs to the same entity. Reducer has 3 primary phases: shuffle, sort and reduce. Input to the Reducer is the sorted output of the mappers. The shuffle and sort phases occur simultaneously.

3. Job Configuration
It represents a Map/Reduce job configuration. It is the primary interface for a user to describe a Map/Reduce job to the Hadoop framework for execution. It is configures the mapper with the linkage task.

4. Job Tracker
Job Tracker fetches input splits from the shared location where the Job Client placed the information. Then creates a map task which performs linkage for each split. Each linkage task in map is assigned to a Task Tracker. The Job Tracker monitors the health of the Task Trackers and the progress of the linkage. As map tasks complete and results become available, the Job Tracker creates reduce tasks up to the maximum enabled by the job configuration. Data linkage is complete when all map and reduce tasks successfully complete, or, if there is no reduce step, when all map tasks successfully complete.

5. Task Tracker
A Task Tracker manages the linkage tasks of one worker node and reports status to the Job Tracker. Often, the Task Tracker runs on the associated worker node. When the Job Tracker assigns a map or reduce task to a Task Tracker, the Task Tracker fetches job resources locally and execute the map or reduce task. Finally reports status to the Job Tracker.

VI. Conclusion
OCCT, a one-class decision tree approach for performing one-to-many and many-to-many data linkage using mapreduce is presented in the paper.

The proposed method is based on a one-class decision tree model that encapsulates the knowledge of which records should be linked to each other. Implementation using mapreduce will reduce the execution time of one-to-many data linkage. It enhances parallelism as linkage is executed in a distributed environment. The proposed method will be very efficient for large datasets. Future work includes extending mapreduce framework for implementing many-to-many datalinkage.

References
[1] Ma'ayan Dror, Asaf Shabtai, Lior Rokach, Yuval Elovici "OCCT: A One-Class Clustering Tree for Implementing One-to-Many Data Linkage", IEEE transactions on Knowledge and Data engineering, Vol. 26, No. 3, March 2014.

[2] A.J. Storkey, C.K.I. Williams, E. Taylor, R.G. Mann,"An Expectation Maximisation Algorithm for One-to-Many Record Linkage", Univ. of Edinburgh Informatics Research Report, 2005.

[3] S. Ivie, G. Henry, H. Gatrell, C. Giraud-Carrier,"A Metric-Based Machine Learning Approach to Genealogical Record Linkage", Proc. Seventh Ann. Workshop Technology for Family History and Genealogical Research, 2007.

[4] P. Christen, K. Goiser,"Towards Automated Data Linkage and Deduplication", Technical report, Australian Nat'l Univ., 2005.

[5] M.D. Larsen, D.B. Rubin,"Iterative Automated Record Linkage Using Mixture Models", J. Am. Statistical Assoc., Vol. 96, No. 453, pp. 32-41, Mar. 2001.

Bargavi C received her B.Tech degree in Information Technology from Anna University, Trichy, India, in 2012, pursuing M.Tech degree in Information Technology from Mepco Schlenk Engineering College, Sivakasi, India. Her research interests include data mining and big data analytics.

Blessa Binolin Pepsi M received her B.Tech degree in Information Technology from Institute of Road and Transport Technology, Erode, India, in 2010, pursuing M.E. degree in Computer Science and Engineering from Mepco Schlenk Engineering College, Sivakasi, India in 2012. Her research interests include data structures and knowledge and data engineering. She is currently working as assistant professor, with the Department of Information Technology, in Mepco Schlenk Engineering College, Sivakasi, India.