Efficient Distributed MST Based Clustering for Recommender Systems

Ahmad Shahzad
School of Electrical Engineering and
Computer Science
University of Liverpool, Liverpool, L69 3BX, U.K
Email: ahmads@liverpool.ac.uk

Frans Coenen
School of Electrical Engineering and
Computer Science
University of Liverpool, Liverpool, L69 3BX, U.K
Email: coenen@liverpool.ac.uk

***Abstract*—This paper presents the Distributed Kruskal Algorithm for Minimum Spanning Tree (MST) based clustering to be used in the context of recommendation engines. The algorithm can operate over large graph data sets distributed over a number of machines. The operation of the algorithm is evaluated by comparing both the quality of the cluster configurations produced, and the accuracy of the predictions, with non-MST based clustering approaches. The results indicate that the proposed approach produces comparable recommendations at much lower storage, hence runtime, costs.**

Keywords*—*Minimum Spanning Tree based Clustering, Recommendation Engines

I. Introduction

Clustering is a an important and very well studied machine learning problem. The aim of clustering is to partition a given data set, possibly a graph data set, into smaller groups to maximise the inter-cluster similarity and minimise the intra-cluster similarity. Clustering has many applications but one application, and that of interest with respect to this paper, is recommendation engines. The idea is to recommend items to the users of a recommendation engine according to the cluster into which they are located. To cluster the users of a recommendation engine the idea presented in this paper is to represent the users and items as a graph where the weights are the strength of the connections between user and items.

There are many graph clustering techniques that have been proposed, but one technique, and that of interest with respect to this paper is *Minimum Spanning Trees* (MST) based clustering. Given a connected, edge weighted, undirected graph $G = (V, E)$, where V is a finite set of nodes, and $E \subseteq V \times V$ is a finite set of edges such that each edge $e_i \in E$ has a weight w_i, a spanning tree T is a subset E such that all the vertices are connected without any cycles. Formally:

$$T = (V, E') \text{ where } E' \subseteq E$$

A graph can have many spanning trees, but all have $|V|$ vertices and $|V|$ -1 edges. The MST for a graph G is the spanning tree with the minimum possible total edge weigh $w(T)$:

$$w(T) = \sum_{e \epsilon E(T)} w(e) \tag{1}$$

The size of the graphs that we wish to process, has increased significantly in the age of digitisation and the era of artificial intelligence. Large graphs appear in a number of contexts including recommendation engine context. These graph data sets cannot be held in the memory typically available with respect to a single machine. Hence, programming paradigms like *Spark* are becoming increasingly popular. Using Spark data is distributed across a cluster of machines; algorithms are then applied over the cluster rather than using a single machine. Consequently, an important area of study is the operation of classical graph algorithms in the context of cluster computing frameworks. One example, and that of interest with respect to the work presented in this paper, is the processes whereby a graph can be mapped onto a cluster of machines. Given n machines in a cluster, E will be distributed uniformly across the cluster so that each machine n_i will hold approximately $|E|/n$ edges. Thus, to derive the MST for a given graph G, it is necessary to find the subset of edges E' that minimises $w(T)$k (as defined in Eq. 1). This paper presents the Distributed Kruskal Algorithm, a distributed MST algorithm inspired by the classical Kruskal Algorithm [1] and the *Prarallel Prim's* algorithm [2] which improve the run time performance compared to earlier approaches.

The rest of this paper is organised as follows. An overview of relevant previous work is presented in Section II. This is followed by a description of the proposed approach in Section III. The evaluation of the approach is given in Section IV. The paper is then completed with some conclusions presented in Section V.

II. Related Work

MST based clustering has been rigorously tried and tested [3]. Traditional MST-based clustering algorithms usually use the Euclidean distance between two end vertices as the edge weight. However, in [4] the cosine similarity was used as the edge weight. Cosine similarity is computed by treating two vertices as a vectors and their relationship with rest of graph nodes as vector dimensions. Successful attempts have been made to improve the efficiency of *MST* based clustering. For example in [5] MST-based clustering using a partition-based based approach over a nearest neighbour graph was proposed for reducing the computational overhead. Whilst in [6] a two level approximate

Euclidean MST approach for efficient calculation of *MST* was proposed.

However, the above algorithms are intended for deployment on single machines. As noted in the introduction to this paper the size of the graph datasets we wish to process are increasing year-on-year; hence it is often the case that graph data has to be distributed across a cluster of machines. In [2] the Parallel Prim's algorithm was proposed, a mechanism for generating an MST given a distributed graph data set. The algorithm assumes that graph vertices can fit into the memory of a single machine whilst graph edges cannot and hence are distributed across a cluster of machines.

MST based clustering techniques have been used in recommendation engines. Typically edge weights are calculated using the Euclidean distance, cosine similarity or dissimilarity. A further alternative is to use collaborative filtering to improve the effectiveness of the recommendation [7].

III. MSF based Clustering

In this section the proposed MST based clustering for recommendation engine approach is presented. The section is divided into two parts, Sub-section III-A presents the proposed MST generation algorithm, the Distributed Kruskal's Algorithm, whilst

A. The Distributed Kruskal's Algorithm

The proposed Distributed Kruskal algorithm is presented in this sub-section. The algorithm is founded on the classical Kruskal's algorithm, and takes inspiration from *Parallel Prim's* algorithm [2] but differs in the way edges are grouped and partitioned over the cluster taking into account the total number of nodes available. The proposed Distributed Kruskal Algorithm, as in the vase of the approach presented in [2], assumes that the memory available at the main node is sufficient for all the vertices to be stored, but that edges have to be distributed over a cluster of machines. The proposed algorithm uses a disjoint set data structure, with parent and rank hash maps. It is assumed that the main node of the cluster has more memory available than the worker nodes so that it can reduce all the data which are processed in parallel by the worker nodes.

The pseudo code for the proposed Algorithm is presented in Algorithm 1. The algorithm takes as input a Graph $G(V, E)$ and the number of nodes n available in the mchine cluster. Then, line 2, it initialises a disjoint set, d, which will use two hash maps; one containing the rank of each vertex and other the parent of each vertex. In line 3 and empty MSF is defined which will be populated and returned. In line 4 an empty list is defined to hold edges leaving *the disjoint set. The loop starting at line 5 populates d and the associated hash maps for each vertex. At line 8, the edges of the graph are distributed across the n available machines in a random manner. Then a while loop is started (line 9) which continues as long as there are any remaining edges which are leaving the disjoint set. At line 10 a hash map is created which contains the parent vertex for each vertex. This newly created hash map is then populated using the disjoint set maintained during the life cycle of the algorithm. The parent map is broadcast to all the nodes at line 14. Line 15 starts a parallel execution of the code on all nodes where the edges on each machine find minimum edges leaving the disjoint set which are then reduced to the main node to form a consolidated set at line 18. A loop starting at line 19 then takes the union of all the edges to the disjoint set one by one, and adds them to the MSF. Once the loop is finished it returns the MSF.

Algorithm 1 Distributed Kruskal's Algorithm

```
 1: function DistributedKruskal($G(V, E), n$)
 2:     $d \leftarrow DisjointSet()$
 3:     $msf \leftarrow \{\}$
 4:     $edgesLeavingDisjointSet \leftarrow E$
 5:     for all $v \in V$ do
 6:         $d.makeSet(v)$
 7:     end for
 8:     $distributedEdges \leftarrow distributeEdges().groupByKey(n)$
 9:     while $|edgesLeavingDisjointSet| \geq 1$ do
10:         $parentMap \leftarrow HashMap()$
11:         for all $v \in V$ do
12:             $parentMap.insert(v, d.find(v))$
13:         end for
14:         Broadcast $parentMap$
15:         for all $e \in distributedEdges$ do    ▷ mapper function executed in parallel
16:             $findMinimum(e, parentMap)$
17:         end for
18:         $edgesLeavingDisjointSet \leftarrow reduceMaps()$
19:         for all $edge(u, v) \in edgesLeavingDisjointSet$ do
20:             $d.union(u, v)$
21:             $msf.insert(edge(u, v))$
22:         end for
23:     end while
24:     return $msf$
25: end function
```

The presented algorithm 1 taken input of a graph $G(V, E)$ and returns and MSF which contains the set V and the selected edges.

B. Clustering Techniques

Data clustering is a common unsupervised machine learning task. There are as range of clustering techniques that have been proposed and frequently used. For the evaluation presented later in this paper two of the most frequently referenced were selected:

- k-Means Algorithm
- Bisecting k-Means Algorithm

Both of these algorithms include in their input the number of required clusters k. However, the k-Means algorithm does not consider the number of items within a cluster, where as Bisecting k-Means attempts to keep the number of data points within each cluster uniform. The relevance here is that data clustering is also a good mechanism for distributing data over a cluster of machines where it is typically important that the data is distributed equally over machines (in other words the data clusters are of similar size). The significance is that an equal distribution of data across a cluster of machines provides for more effective "data look up". So both of the above mentioned algorithms are also evaluated in terms of standard deviation of cluster sizes.

IV. Evaluation

This section presents the results of a sequence of experiments conducted to evaluate the proposed approach. For the experiments the Movielens database was used, The *Movielens* database [8] is a popular choice for the study of recommendation engine algorithms. Movielens is a web-based recommender system that can be used to recommend movies to its members according to their movie preferences. It operates by applying collaborative filtering to its user's movie ratings and reviews. It contains about 1 million reviews for 3952 movies. Rating are expressed using the numeric range 1 to 5. For the evaluation presented in this section 1 million ratings, provided by 6040 users, were used.

For the evaluation both the quality of the cluster configuration produced using the proposed MST based approach and the quality of the recommendations were considered. For the first experiments were conducted using the two clustering algorithms described in Sub-section III-B: (i) k-Means and (ii) Bisecting k-Means. The quality of the cluster configurations produced using the proposed MST approach (MST Data), using both k-Means and Bisecting k-Means, was compared with the quality of the cluster configurations produced when k-Means or Bisecting k-Means was applied to the the whole Movielens dataset (Full Data). Two mechanisms for measuring the quality of a cluster configuration were used: (i) *Silhouette Score* (coefficient) and (ii) *Cosine Similarity Score*. A range of values for k were considered, $k = \{10, 20, 50, 100\}$. Recall that the MST Data, produced using the proposed Distributed Krusal's MST algorithm, will be a lot smaller than the original Full Data. The MST data contains all the vertices of the full data set, but only the edges that are part of MST.

A good cluster configuration does, of course, not necessarily mean that good recommendations are made. To test the quality of the recommendations the data sets (MST Data and Full Data) were split into a training and test component using a ration of 90 : 10. The accuracy with which the top ten movies were predicted was then recorded. Again both k-Means and Bisecting k-Means clustering was used applied to both the MST Data and the Full Data, however, in this case the range of values for k was $k = \{10, 20, 50, 100, 300\}$

All the experiments were conducted on a spark cluster. There were 5 spark nodes in total, One of them is main driver node and other 4 were worker nodes. Each worker node had 2 GB of memory and Spark Driver had 4 GB of memory available. Each node had an i7 quad core 2.6HZ processor. All nodes were connected via hadoop-yarn.

The remainder of this section is organised as follows. The two cluster configuration evaluation metrics used are presented in Sub-section IV-A. The results obtained with respect to the cluster configuration analysis are given in Sub-section IV-B, while the results for the recommendation accuracy are given in Sub-section IV-C.

A. *Cluster Configuration Evaluation*

The silhouette score of a cluster is an indicator of both the cohesiveness and separation of a cluster configuration [9]. For any data points i in a cluster C_i, the mean distance $a(i)$ between the point i and all other data points in the same cluster can be defined as:

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{2}$$

where $d(i, j)$ is the distance between data points i and j.

For any data points i in a cluster C_i), the minimum mean dissimilarity $b(i)$ between the point i and the data points in any other cluster C_k can be defined as:

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{3}$$

The Silhouette Score of a data point i is then given by:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, if |C_i| > 1 \tag{4}$$

The overall Silhouette Score for a given cluster configuration is then the mean of the individual point scores. Silhouette Scores ranges from -1 to 1, where a higher score indicates a good cluster configuration

The Cosine Similarity Score [10] is a measure of the cohesiveness of the clusters in a cluster configuration. Given two points described by two vectors, A and B, the Cosine Similarity Score is calculated as follows:

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} \mathbf{A}_i \mathbf{B}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{A}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{B}_i)^2}} \tag{5}$$

The range of *Cosine Similarity Score* is from 0 to 1. Hence, two points represented by identical vectors will have a Cosine Similarity Score of 1. To determine the quality of a cluster configuration, Cosine Similarity Score for each user within a cluster is calculated and summed up to represent the score for that cluster.

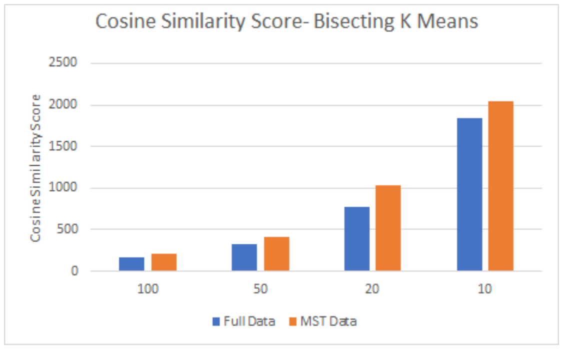

Figure 1: Cosine Similarity for bisecting k-means

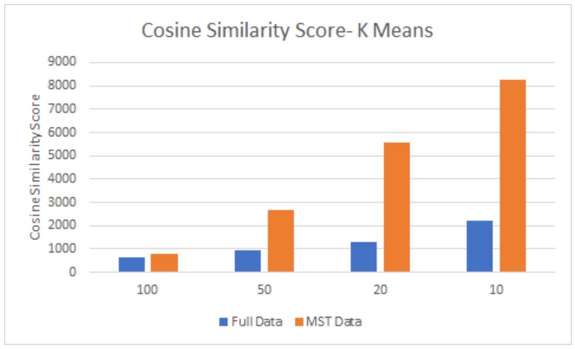

Figure 2: Cosine Similarity for k-means

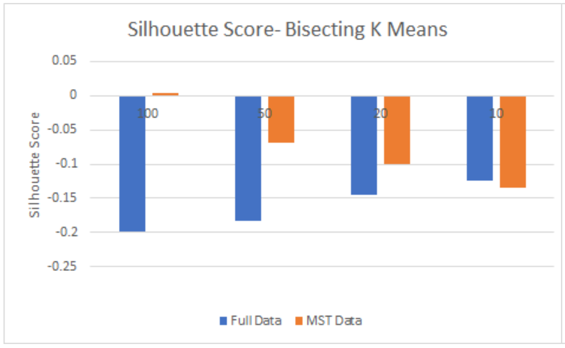

Figure 3: Silhouette Score bisecting k-means

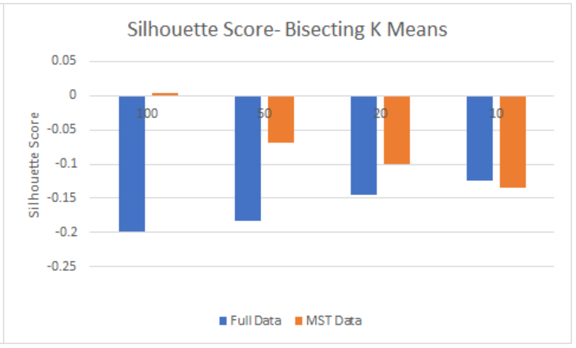

Figure 4: Silhouette Score k-means

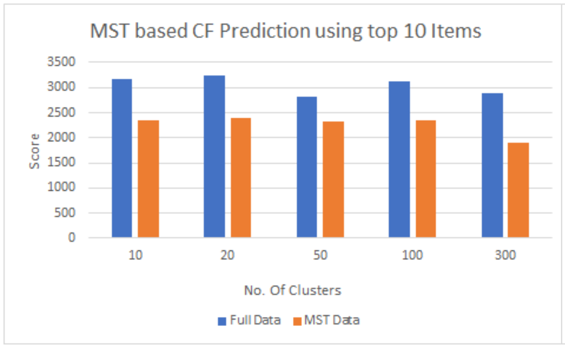

Figure 5: Prediction Using Top 10 Items

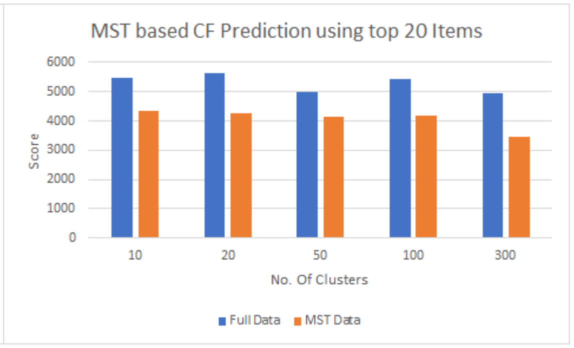

Figure 6: Prediction Using Top 20 Items

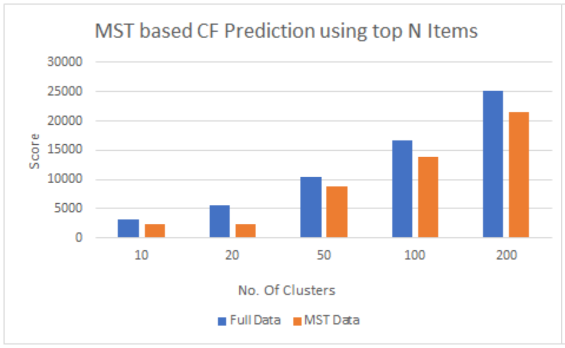

Figure 7: Prediction Using Top N Items

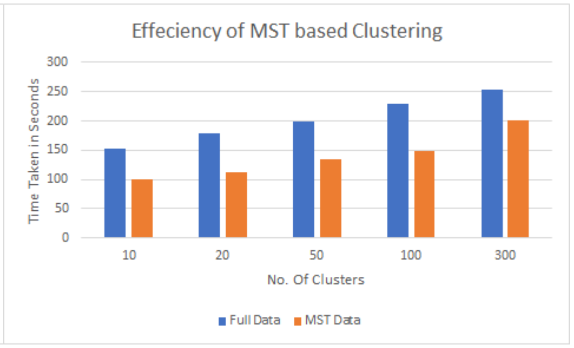

Figure 8: Efficiency of MST based Clustering

B. Cluster Configuration Analysis

The results for the cluster configuration quality analysis are presented in Figures 1 to 4. In the figure the x-axis represents k and the y-axis the Silhouette or Cosine Similarity Score as appropriate. Figures 1 and 2 give the Cosine Similarity Scores, and Figures 3 and 4 the Silhouette Scores. From the figures it can be seen that best Cosine Similarity Scores are obtained using low values of k, and that better scores are obtained when using the proposed Distributed Krusal's MST algorithm (the mST Data). The best Cosine Similarity Score was obtained sing the MST Data, $k = 10$ and k-Means clustering. With respect to the Silhouette scores many negative values were recorded indicating that some samples might have been assigned to the wrong cluster. Interestingly, the best Silhouette scores when using the MST Data were obtained using high values of k, although not the case when using Full Data.

C. Recommendation Analysis

The results for the recommendation analysis are presented in Figures 5 to 8. In the figure the x-axis represents k and the y-axis the number of movies correctly recommended. From the figures it can firstly be seen that with respect to both data sets the accuracy scores are consistent, regardless of cluster sizes. It can also be seen that when using the Full data a better recommendation accuracy is obtained than when using the MST Data. However, the advantage offered by the MST based approach is that it is more efficient because the MST Data is much smaller. To calculate MST Data over one million data set of Movielens it takes less than 10 seconds using Distributed Kruskal's algorithm.

V. Conclusions and further suggestions

In this paper an approach to MST-based clustering for recommendation engines has been presented. The idea was, that by using the proposed MST-based approach, the data would be reduced in size so that it comprised the set of vertices V and only the minimum number of edges from the set of edges E. The resulting MST data could then be clustered to support recommendation. For MST generation the Distributed Kruskal algorithm was proposed. The approach was evaluated with respect to two clustering algorithms, k-Means and Bisecting k-Means, and comparisons made between using the MST Data and the Full Data. It was found that the proposed MST-based approach produced better cluster configurations than when using the Full Data, however, at the expense of slightly reduced recommendation prediction accuracy. However, the overall advantage was that the proposed approach would be much faster than alternative Full Data approaches. Hence, the proposed approach is suited to settings where speed and efficiency needs to be preferred over accuracy, like in matchmaking for on-line games. For future work, the authors intend to apply their approach to further data sets so as to strengthen the empirical analysis presented in this paper.

References

[1] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *AMS*, vol. 7, pp. 48–50, 1956.

[2] R. P. Swaroop Indra Ramaswamy, "Distributed minimum spanning trees," 2015. [Online]. Available: http://stanford.edu/~rezab/classes/cme323/S15/projects/distributed_minimum_spanning_trees_report.pdf

[3] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers*, vol. C-20, no. 1, pp. 68–86, 1971.

[4] A. Huang, "Similarity measures for text document clustering," *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, 01 2008.

[5] R. Jothi, S. K. Mohanty, and A. Ojha, "Fast approximate minimum spanning tree based clustering algorithm," *Neurocomputing*, vol. 272, pp. 542 – 557, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121731295X

[6] X. Wang, X. Wang, and X. Li, *A Fast Two-Level Approximate Euclidean Minimum Spanning Tree Algorithm for High-Dimensional Data*, 07 2018, pp. 273–287.

[7] O. Baida, N. Hamzaoui, A. Sedqui, and A. Lyhyaoui, "Recommendation based on co-similarity and spanning tree with minimum weight," 09 2012, pp. 355–359.

[8] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context." *ACM Transactions on Interactive Intelligent Systems*, no. 4, 2015. [Online]. Available: https://dl.acm.org/doi/10.1145/2827872

[9] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53 – 65, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377042787901257

[10] M. Schwarz, M. Lobur, and Y. Stekh, "Analysis of the effectiveness of similarity measures for recommender systems," in *2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)*, 2017, pp. 275–277.