IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 23, 2019, accepted October 18, 2019, date of publication October 24, 2019, date of current version November 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949455

Distributed Clustering of Text Collections

JUAN ZAMORA[1], HÉCTOR ALLENDE-CID[2], AND MARCELO MENDOZA[3]
[1]Instituto de Estadística, Pontificia Universidad Católica de Valparaíso, Valparaíso 2340023, Chile
[2]Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile
[3]Centro Científico y Tecnológico de Valparaíso, Universidad Técnica Federico Santa María, Valparaíso 2390123, Chile

Corresponding author: Juan Zamora (juan.zamora@pucv.cl)

The work of J. Zamora was supported by the Postdoctoral Project (Nr. 3180689) funded by CONICYT-FONDECYT, Government of Chile. The work of H. Allende-Cid was supported by the Project (Nr. 11150248) funded by CONICYT-FONDECYT, Government of Chile. The work of M. Mendoza was supported in part by PIA/Basal FB0821-CONICYT and in part by the Millennium Institute for Foundational Research on Data, Government of Chile.

ABSTRACT Current data processing tasks require efficient approaches capable of dealing with large databases. A promising strategy consists in distributing the data along with several computers that partially solve the undertaken problem. Finally, these partial answers are integrated to obtain a final solution. We introduce distributed shared nearest neighbors (*D-SNN*), a novel clustering algorithm that work with disjoint partitions of data. Our algorithm produces a global clustering solution that achieves a competitive performance regarding centralized approaches. The algorithm works effectively with high dimensional data, being advisable for document clustering tasks. Experimental results over five data sets show that our proposal is competitive in terms of quality performance measures when compared to state of the art methods.

INDEX TERMS Distributed algorithms, distributed text clustering, high dimensional data.

I. INTRODUCTION

As a consequence of the explosive growth of the web, the integration of search engines into personal computers and mobile devices, and the extensive use of social networks, the clustering of text for document organization has become a crucial aspect for web data management. Clustering is one of the most critical tasks in text mining. It plays an essential role in efficient document organization, topic extraction, summarization, and ad-hoc information retrieval. Nowadays, the generation of large amounts of documents surpasses the computational capacity of personal computers and even one of the high-performance computers. Recent estimates indicate that the amount of web pages indexed in the web is higher than 50 billion.[1] Therefore, it is of great interest to develop algorithmic techniques able to organize, classify, and summarize document collections. Accordingly, it is necessary to implement algorithms that work with text collections distributed into multiple machines. Also, these algorithms have to perform efficiently in modern hardware, particularly within parallel processing frameworks in multi-core architectures.

In real scenarios such as *Collection Selection* [4] for distributed search engines, in which for a given query the node containing the most suitable collection must be picked, the challenges related to scalability and efficiency of clustering methods have become very important [24]. Traditional algorithms often assume that the whole dataset fits main memory (RAM), and thus, every document can be accessed at any time with no access latency. In scenarios where the size of the collection is much bigger than RAM, this memory allocation process is unfeasible.

There are two approaches successfully applied to the construction of clustering algorithms able to process big data. The first one introduces constraints on the number of passes allowed on a document [32]. These types of algorithms involve less computational costs than other methods, although they tend to produce large clusters. Also, these methods provide groups that have a strong dependence on the order in which the data is processed. The second approach exploits multi-core architectures to perform parallel processing of the data [33]. Talia [28] identified three primary strategies in the parallelism used in data mining algorithms. 1) Independent parallelism where each processor accesses to the whole data to operate but do not communicate with each other. 2) Task parallelism where each processor runs different tasks in each partition or the entire data set. 3) Single Program Multiple Data, or SPMD parallelism, where multiple processors execute the same algorithm on different partitions and exchange the partial results to cooperate. Most of the

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaojie Guo.

[1]http://www.worldwidewebsize.com/, last access at 26th June 2019

parallel clustering algorithms follow the combinations of task and SPMD parallelism using a master-slave architecture. SPMD combines the computational power of a single machine with the scalable storage capability of a distributed system by partitioning the data into several independent nodes connected through a network [27].

In general, SPMD algorithms work at two levels: local level, and global level [17], [30], [31]. At the local level, all nodes carried out a clustering algorithm independently from the other nodes. Usually, the same clustering algorithm is running in each node, producing a collection of local models. Then, these models are transferred to a central node to form a global model. The global model is then transmitted to nodes to update the local models [31]. We adhere to this strategy to design our algorithm.

SPMD seems promising since it allows us to exploit the local capabilities of single computers with multi-core architectures. Within this research line, there are two variants regarding the data generation scenario. On the one hand, in some problems where the data set is extensive but collected in a centralized fashion, the strategy employed consists in partitioning the collection into several nodes. This scheme leads to the transmission of a lot of data during the execution of the algorithm [26]. On the other hand, there are some problems inherent to distributed databases as privacy issues [13] or costs involved in data consistency constraints [21].

The focus of our work is to provide a clustering algorithm capable of working with text collections. Text collections, given how they are represented, lead to high-dimensional vector representations. An effective way to cluster high-dimensional data is to use the shared nearest neighbors (SNN) algorithm [15]. SNN can process data according to a notion of density, which works well with high dimensional data.

Ravichandran *et al.* [36] introduce a modification of SNN to work with high dimensional data. It deals with the hubness problem, i.e. how to discard the effect of highly connected points in density estimation. The authors use a transformation called unscented transform to sample the core points of the SNN space. The transformation approximates a Gaussian, from which a density sampling is performed on the candidate points. The work shows that this modification in the density sampling method could improve the robustness to noise and, at the same time, limit the oversampling of hubness points. The effects of this modification would indicate that the algorithm could make a better estimate of the number of clusters since it would limit the effect of noise and outliers in the clustering process.

We take some inspiration from Ravichandran's work to use density-based sampling in an environment with distributed data. Since density-based sampling shows good properties to noise and outliers, we decided to study its properties on disjoint text partitions. As a result, we introduce Distributed Shared Nearest Neighbors clustering (*D-SNN* for short), an algorithm that can work with distributed text datasets. *D-SNN* is an extension of the *C-SNN* clustering algorithm introduced by Ertoz *et al.* [10] that works with centralized data. Our algorithm works over disjoint data partitions, conducting sampling over *core points* in each node. *Core points*, previously introduced in the famous algorithm *DBSCAN* [9], are density prototypes. Our algorithm takes advantage of *core points* using them to conduct density-based sampling at the node level. Then, a master node collects each sample to conduct a consolidation phase, producing a global cluster solution. To validate our proposal, we conduct experiments over five data sets, showing that our algorithm is feasible and outperforms *C-SNN*. The use of *core points* as density prototypes makes *D-SNN* resistant to white noise. This property is a skill that *D-SNN* maintains since it uses density-based sampling, just like *DBSCAN*. Since *D-SNN* searches for high-density data regions, it can discard low-density regions with white noise. However, the detection of *core points* introduces two parameters, Eps and MinPts, the similarity threshold and the minimum number of points in each data ball needed to tag core points, respectively. To limit the effect of parameter sensibility on cluster quality, we provide a data-driven approach for parameter tuning.

To achieve this result, we explored several research directions.

- We evaluate clustering algorithms in five different datasets, showing that *D-SNN* outperforms its competitors.
- We study the time complexity of *C-SNN* density-based clustering.
- We study how to use density-based sampling favoring resistance to white noise in low-density regions.
- We study how to provide a data driven approach for parameter tuning.

By addressing all these problems, we show that it is possible to provide a distributed clustering algorithm to deal with text collections. The main strength of the proposed algorithm is its ability to use disjoint data partitions to return high-quality clustering results. This strength will avoid centralizing data partitions in a single data node to provide a clustering overview, reducing the computing load for clustering distributed data partitions.

This article extends our prior contribution:

- *D-SNN*, suitable for distributed clustering in text collections (previously introduced in [40]).

Novel, unpublished contributions of this article are:

- a complete description of *D-SNN*, including diagrams, pseudocode, and illustrative examples;
- a discussion about the computation complexity of *D-SNN*;
- a new set of tests, based on Tipster datasets (DOE, ZF, FR and SJMN);
- a new data-driven parameter tuning method, including examples on Tipster datasets;
- results on computational load and feasibility of *D-SNN* on distributed data partitions.

The remainder of this document is structured as follows. First, a review of the literature on scalable and distributed

data clustering methods is presented in Section II. Next, we introduce our algorithm in Section III. Experimental design, along with the attained results, is presented and discussed in Section IV. Finally, we conclude in Section V highlighting conclusions and discussing future work.

II. DISTRIBUTED CLUSTERING ALGORITHMS

As far as we know from the literature, most of the existing efforts for the construction of clustering techniques capable of operating in scenarios with distributed data have been focused on low dimensional data (less than 100 attributes) in contrast with document collections in which a document vector for a small collection may have about 10^4 attributes. Nevertheless, the main advances in distributed data clustering are detailed below, especially highlighting those contributions focused on methods capable of dealing with high dimensional data.

A. PARALLEL CLUSTERING ALGORITHMS

Xu *et al.* [31] presented a parallel version of DBSCAN (PDBSCAN). The authors introduced the 'shared-nothing' architecture with multiple computers interconnected through a network. A fundamental component of the shared-nothing system is its distributed data structure. They introduced the dR*-tree, a distributed spatial index structure in which the data is spread among multiple computers and the indexes of the data are replicated on every computer. Using this index, the authors proposed a parallel version of DBSCAN which could run simultaneously on several data partitions. Performance evaluation showed that PDBSCAN offers nearly linear speed-up and an excellent scale-up and size-up behavior. Dhillon and Modha [7] presented an algorithm that exploits the inherent data-parallelism in the k-means algorithm. They analytically showed that the speed-up and the scale-up of the algorithm approach the optimal as the number of data points increases. The implementation of this proposal was done on an IBM POWER parallel SP2 with 16 nodes. The authors showed that the algorithm achieved nearly linear relative speed-ups. Also, the algorithm achieved linear scale-up in the size of the data set and the number of clusters desired. For a 2-gigabyte test data set, the implementation drives at more than 1.8 Gigaflops.

Another scalable approach based on secondary memory consists in designing algorithms able to work within the MapReduce framework. In this context we highlight the contributions made by Ene *et al.* [8] in which they tackle the k-medians problem by using MapReduce. The authors proposed a fast clustering algorithm with constant-factor approximation guarantees. The algorithm uses sampling to decrease the data size, running a time-consuming clustering algorithm such as local search or Lloyd's algorithm on the resulting data set. The proposed algorithm has sufficient flexibility to be used in practice since they run in a constant number of MapReduce rounds. The experiments show that the algorithms' solutions are similar to or better than other solutions.

Bahmani *et al.* [2] introduced a novel parallel implementation of k-means. The proposed algorithm, named k-means++, obtains an initial set of centers that is close to the optimum solution. A major limitation of k-means++ is its sequential nature, which avoids its use on massive data. This limitation is due to one must make k passes over the data to find a good initial set of centers. Then, the authors show how to drastically reduce the number of passes needed to obtain a good initialization. The proposed initialization obtains a nearly optimal solution after a logarithmic number of passes and then shows that in practice a constant number of passes suffices.

De Vries *et al.* [6] proposed a scalable algorithm that clusters hundreds of millions of web pages into hundreds of thousands of clusters. It does this on a single mid-range machine using efficient algorithms and compressed document representations. The algorithm was applied to two web-scale crawls (ClueWeb09 and ClueWeb12) covering tens of terabytes of text. The proposed algorithm uses the entire collection in clustering and produces several orders of magnitude more clusters than the existing algorithms. Fine-grained clustering is necessary for meaningful solutions in massive collections where the number of distinct topics grows linearly with collection size. Cluster quality was assessed in two downstream tasks, the ad-hoc search of relevance judgments and spam classification. These results show that the clustering on massive data is useful for a number of web data management tasks.

A parallel version of SNN was proposed by Kumari *et al.* [37] using an R-tree structure. The parallel version makes use of an implementation of the R-tree in shared and distributed memory. The focus of the work is on the efficiency of the algorithm, so the authors provide evidence that indicates that its implementation is scalable in massive datasets. Although they do not provide evidence regarding the use of the algorithm in high dimensional data, since the dataset of greater dimensionality used by the authors is KDDCUPB74d, with vectors of 74 components, the experiments show very competitive speed-ups in massive datasets. A limitation of the work consists in the lack of evidence concerning the quality of the clusters found.

Plattel [39] proposed a distributed version of SNN. In that version, a centralized data structure is maintained from which the distribution of data to different machines is carried out. That version of SNN requires the choice of a machine that synchronizes the process of data distribution as well as the creation of a structure that indexes the collection. The authors indicate that it is enough to use a decision tree to index the data. The tree is used to drive the clustering process helping the phase of merging in the master node.

B. CLUSTERING ON DISTRIBUTED DATA

Kargupta *et al.* [16] proposed a method to conduct Principal Component Analysis (PCA) over heterogeneous and distributed data. Based on this contribution, they also introduced a clustering method. Once the principal global components

are obtained by using the distributed method, they are transmitted to each node. In each partition, the local data is projected onto the components, and then a traditional clustering technique is applied. Finally, a master node integrates the local clusters to obtain a global clustering solution.

Liang *et al.* [20] presented another algorithm for PCA over distributed data. In that proposal, each node computes PCA over its local data and transmit them to a master node. This node uses the received components to estimate the principal global components, which are then sent to each data node. Then, in every node, each data partition is projected onto the global components to compute a coreset. Finally, the global coreset is used to obtain a global clustering model.

Li *et al.* [19] proposed an algorithm named CoFD, which is a non-distance based clustering algorithm for high dimensional data. Based on the Maximum Likelihood Principle, CoFD attempts to optimize its parameter settings to maximize the likelihood between data points and the model generated by the parameters. Then, a distributed version of the algorithm, called D-CoFD, was proposed. The authors claim that the experimental results on both synthetic and real data sets show the efficiency and effectiveness of both algorithms.

C. DISTRIBUTED APPROACHES FOR DENSITY-BASED CLUSTERING

Klusch *et al.* [17] proposed a novel distributed clustering algorithm based on non-parametric kernel density estimation, which takes into account the issues of privacy and communication costs that arise in a low dimensional distributed data environment. Januzaj *et al.* [14] presented a scalable version of DBSCAN that is also capable of operating over distributed collections. First, the best local representatives are selected depending on the number of points that each one represents sending the chosen points to a master node. The master node is in charge of clustering the local representatives into a single new model, which is then transmitted to the data nodes improving their local group structure. Unfortunately, experimental results show that the algorithm does not work well with high dimensional data.

D. DISTRIBUTED APPROACHES FOR PARAMETRIC CLUSTERING

Merugu and Ghosh [25] introduced a framework for clustering distributed data in unsupervised and semi-supervised scenarios, taking into account several requirements, such as privacy and communication costs. Instead of sharing samples of the original data, the authors transmit the parameters of suitable generative models built in each data node to a master node. The authors show that the best representative of all the data is a kind of "average" model. They show that this model can be approximated by sampling the underlying local distributions using Markov Chain Monte Carlo techniques. Also, a new measure that quantifies privacy based on information-theoretic concepts was proposed, showing that decreasing privacy leads to a higher quality of the combined model. The authors provide empirical results on different data types to highlight the generality of their framework. The results show that high quality distributed clustering can be reached with little privacy loss and low communication cost.

Kriegel *et al.* [18] proposed a distributed model-based clustering algorithm that uses the EM algorithm for detecting local models in terms of mixtures of Gaussian distributions. They present an efficient and effective algorithm for deriving and merging local Gaussian distributions producing a meaningful global model. In a broad experimental evaluation they demonstrate that the framework scales-up in a highly distributed environment.

E. DISTRIBUTED APPROACHES FOR PROTOTYPE-BASED CLUSTERING

Forman and Zhang [11] introduced a technique to parallelize a family of centroid-based data clustering algorithms. The idea of the proposed algorithms is to communicate only sufficient statistics, yielding linear speed-up with excellent efficiency. The proposed technique does not involve approximation and may be used in conjunction with sampling or aggregation-based methods, such as BIRCH [23], lessening the degradation of their approximation and handling large data sets. The authors demonstrate that even for relatively small problem sizes, it can be more cost-effective to cluster the data in-place using an exact distributed algorithm than to collect the data in one central location for clustering.

Balcan *et al.* [3] provides novel algorithms for distributed clustering for two popular prototype-based strategies, k-median, and k-means. The proposed algorithms have algorithmic guarantees and improve communication complexity over existing approaches. The proposed algorithm maps the problem of finding a clustering solution to finding a small size coreset. The authors provide a distributed method for constructing a global coreset which improves over the previous methods by reducing the communication complexity, and which works over general communication topologies. Experimental results on large scale data sets show that this approach is feasible.

Mashayekhi *et al.* [22] proposed GDCluster, a fully distributed clustering algorithm capable of dealing with dynamic data. GDCluster works by merging cluster prototypes obtained in each node employing a partition or density-based algorithm. In the case of partition-based methods, the prototypes correspond to centroids. For density-based methods, they use core points. GDCluster presents an interesting decentralized way of computing a summarized view of the complete dataset through gossip-based continuous cooperation. Its relation with our proposal corresponds to the usage of DBSCAN-based core points as representatives of each data node. Nevertheless, no experiments over high dimensional data were presented by the authors.

Azimi and Sajedi [1] followed a similar strategy to GDCluster introducing GDSOM-P2P. The method uses a variation of the Self-Organizing Maps (SOM) to select representative data in each data node, sharing this information according to a gossip-based protocol. Besides, GDSOM-P2P

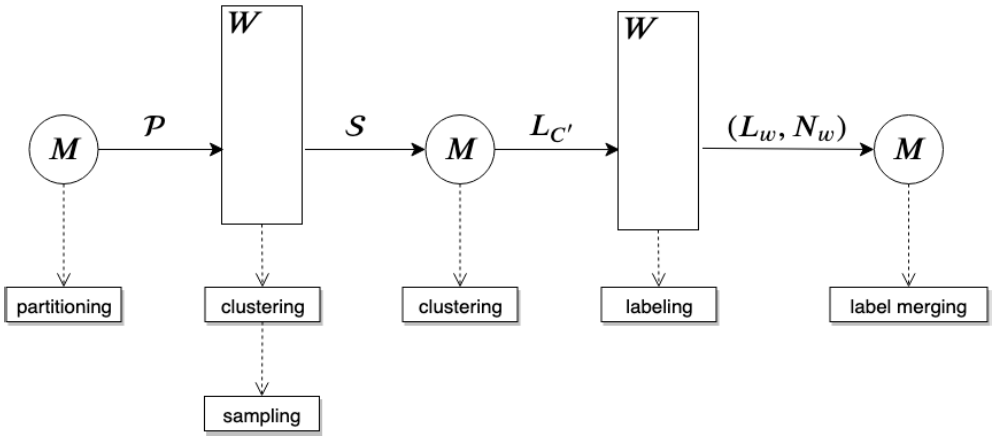

FIGURE 1. Overall scheme for the two stages of the proposed algorithm. $\mathcal{P}$ represents the set of data partitions distributed across workers.

is a robust decentralized algorithm; its dependency on SOM makes difficult its scalability for high dimensional data.

III. DISTRIBUTED SHARED NEAREST NEIGHBORS CLUSTERING

We present a distributed clustering algorithm based on *Shared-Nearest-Neighbor* (SNN) clustering [10]. This method automatically identifies the number of underlying groups in each data partition along with a set of representative points for each one. We pose that this method can deal with collections arbitrarily distributed across a Master/Worker architecture. The algorithm operates in two stages: The first one starts by processing the data in each partition, producing a set of density-based samples, i.e., *core points*, transmitting back a sample set of these *core points* to the master node. In the second stage, the master node joins the sample points received and then conducts an SNN clustering over them. Finally, a set of representative points is labeled, also defining a new set of *core points* that summarizes the overall collection. An overall scheme of the method is shown in Figure 1. The circle symbolizes the master node, and each rectangular block denotes the collection of workers. The continuous arrows represent the communication flow and the labeled boxes the tasks executed by the master or the workers.

A. PARAMETERS OF THE ALGORITHM

The proposed method comprehends four parameters, namely k, and then we could define different values for the parameters and ω. Note that the values of Eps and MinPts may differ between the master and the workers. Accordingly, we could define different values for these parameters having Eps_m and MinPts_m in the master node and Eps_w and MinPts_w in the workers. The value of k represents the size of the neighborhood over which the SNN similarity measure is computed. The neighborhood of each data point is calculated using a proximity function in the original feature space (e.g., term vector space in the case of documents). In the SNN space, the value of Eps denotes the similarity threshold beyond which two points are considered as close. Then, the similarity in the SNN space corresponds to the number of shared neighbors between two different data points. Additionally, a point is identified as a *core point* when the number of points close to it in the SNN space surpasses the value of the density parameter MinPts. Finally, the value of the parameter η rules the size of the sample set of *core points* that is transmitted to the master node.

B. STAGES OF THE ALGORITHM

The overall procedure performed by the master node is depicted in Algorithm 1. Firstly, the dataset is randomly partitioned in a set $\mathcal{P}$ of partitions with $|\mathcal{W}|$ data blocks, assigning each one of them to a worker. Then, the master node is responsible for consolidating the partial results obtained in each worker.

The initial stage starts by performing algorithm 2 in each worker. This procedure starts by computing the shared-nearest-neighbor similarity [15] between each pair of data points and storing it into the matrix **SNN**. This matrix is square and contains as many rows as points the dataset has. Each coefficient $\mathbf{SNN}(i, j)$ denotes the size of the intersection between the k-neighborhoods of two points i and j. In order to deal with high dimensional data, we employ the cosine similarity as the base measure. Note that this function can be replaced with a different one depending on the nature of the data. Then using the **SNN** matrix already computed, each worker runs a centralized shared-nearest-neighbor clustering [10], obtaining a set of core points and clusters. It is after the complete execution of this procedure that the master node receives the core points identified in each worker. Since our method is based on DBSCAN, it may identify too many core points in each cluster, especially under the presence of dense groups. Therefore, only a weighted sample of points from each cluster is transmitted to the master, limiting the costs involved in data communication. The size of the sample is controlled by η.

To explain how this sample is built, let us consider a single worker and a cluster l from a clustering solution $L_{\mathcal{X}}$. A weight is computed for each data point in l, according to the expression:

$$\frac{1-(N_l/N_c)}{2\cdot N_l},$$

where N_c denotes the total number of core points in the clustering solution and N_l the number of core points in l. Note that N_l corresponds to $|l \cap C_{\mathcal{X}}|$, where C_x is the set of core points of the data partition. This function was designed to build a sample that represents both dense and sparse clusters alike avoiding bias to dense clusters. On the one hand, if l is dense, the fraction $\frac{N_l}{N_c}$ will be close to 1, so the weight of each point in l will be small. On the other hand, if N_l is small concerning N_c, the weight of each data point in l will be higher. Finally, in this stage, each worker reports to the master node the sample built from each cluster found.

Algorithm 1 Procedure Executed by the Master Node

Data: $\mathcal{D}$: dataset, $\mathcal{W}$: set of workers, η: sampling fraction
Result: Clustering solution and noise points
Function `master_procedure` ($\mathcal{D}$, $\mathcal{W}$, η, Eps_m, MinPts_m, Eps_w, MinPts_w, k) **:**
- $\mathcal{P} \leftarrow \pi_1 \cup \pi_2 \ldots \pi_{|\mathcal{W}|}$ uniformly drawn from $\mathcal{D}$
- $\mathcal{S} \leftarrow \{\}$
- **foreach** $w \in \mathcal{W}$ **do**
  - $S_w \leftarrow$ `stage1`$(\pi_w, \eta, \mathsf{Eps}_w, \mathsf{MinPts}_w, k)$
  - $\mathcal{S} \leftarrow \mathcal{S} \cup S_w$
- **SNN** $\leftarrow$ `snn_similarity`$(\mathcal{S}, k)$
- $C', L' \leftarrow$ `C_SNN`(**SNN**, Eps_m, MinPts_m)
- $L_{C'} \leftarrow []$
- **foreach** $l \in L'$ **do** `/* pick core points */`
  - append $\{l \cap C'\}$ **to** $L_{C'}$
- $L \leftarrow L_{C'}$
- $N \leftarrow \emptyset$
- **foreach** $w \in \mathcal{W}$ **do**
  - $L_w, N_w \leftarrow$ `stage2`$(\pi_w, L_{C'}, \mathsf{Eps}_w, k)$
  - $N \leftarrow N \cup N_w$
  - **foreach** $l \in L_w$ **do** `/* adding to the overall cluster */`
    - find $j \in [1, |L|]$ s.t. $(L[j] \cap l) \neq \emptyset$
    - $L[j] \leftarrow L[j] \cup l$
- **return** L, N

The master procedure (see Alg. 1) continues after all workers transmit their data point samples to the master node, merging them into a single set $\mathcal{S}$. After calculating the shared-nearest-neighbor similarity on $\mathcal{S}$, the centralized shared-nearest-neighbor clustering procedure is applied obtaining a new set of core points $\mathcal{C}'$ and a global clustering solution L'. Then, the master node calculates a new set of data points, $L_{C'}$,

Algorithm 2 Stage 1 – Worker Procedure

Result: Set of corepoints and a set with sample data from each group
Function `stage1` ($\mathcal{X}$, η, Eps_w, MinPts_w, k) **:**
- **SNN** $\leftarrow$ `snn_similarity`$(\mathcal{X}, k)$
- $C_{\mathcal{X}}, L_{\mathcal{X}} \leftarrow$ `C_SNN`(**SNN**, Eps_w, MinPts_w)
- $S_w \leftarrow \{\}$
- **foreach** $l \in L_{\mathcal{X}}$ **do**
  - $S_l \leftarrow$ sample $\eta * |l|$ points from l using: $\frac{1-(|\{l \cap C_{\mathcal{X}}\}|/|C_{\mathcal{X}}|)}{2\cdot|\{l \cap C_{\mathcal{X}}\}|}$
  - $S_w \leftarrow S_w \cup \{S_l\}$
- **return** S_w

Algorithm 3 Stage 2 – Worker Procedure

Result: Groups of data points assigned to the worker and points identified as noise
Function `stage2` ($\mathcal{X}$, $L_{\mathcal{C}'}$, Eps_w, k) **:**
- SNN $\leftarrow$ `snn_similarity`$(\mathcal{X}, k)$
- $N_w \leftarrow \emptyset$
- $L_w \leftarrow L_{\mathcal{C}'}$
- **foreach** $x \in \mathcal{X}$ **do**
  - $s^* \leftarrow 0$
  - $i^* \leftarrow \emptyset$
  - **for** $i = 1$ **to** $|L_{\mathcal{C}'}|$ **do**
    - $s \leftarrow \max_{q \in L_{\mathcal{C}'}[i]} \mathsf{SNN}(x, q)$
    - **if** $s > s^*$ **then**
      - $s^* \leftarrow s$
      - $i^* \leftarrow i$
  - **if** $s^* \geq \mathsf{Eps}_w$ **then**
    - $L_w[i^*] \leftarrow L_w[i^*] \cup \{x\}$
  - **else** `/* mark as noise */`
    - $N_w \leftarrow N_w \cup \{x\}$
- **return** L_w, N_w

which corresponds to the core points of each cluster identified in the global solution, L'. Then, a consolidated clustering solution L is initialized with $L_{C'}$. After transmitting $L_{C'}$ to each worker, the final stage starts running algorithm 3. Each worker w executes the algorithm 3 using its original data partition π_w. Within this procedure, each data point is associated with the cluster of its nearest core point in $L_{C'}$, excluding those points whose proximity to its nearest core point in $L_{C'}$ is lower than Eps_w. These last points are marked as noise points and added to a set of noise points N_w. Then, the set of noise points N_w, and the list of clusters L_w consolidated in each worker are sent to the master node for a final labeling step.

In the final part of the algorithm 1 the local clusters L_w and the noise sets N_w reported by the workers are merged into the global clustering solution. Finally, the overall clustering

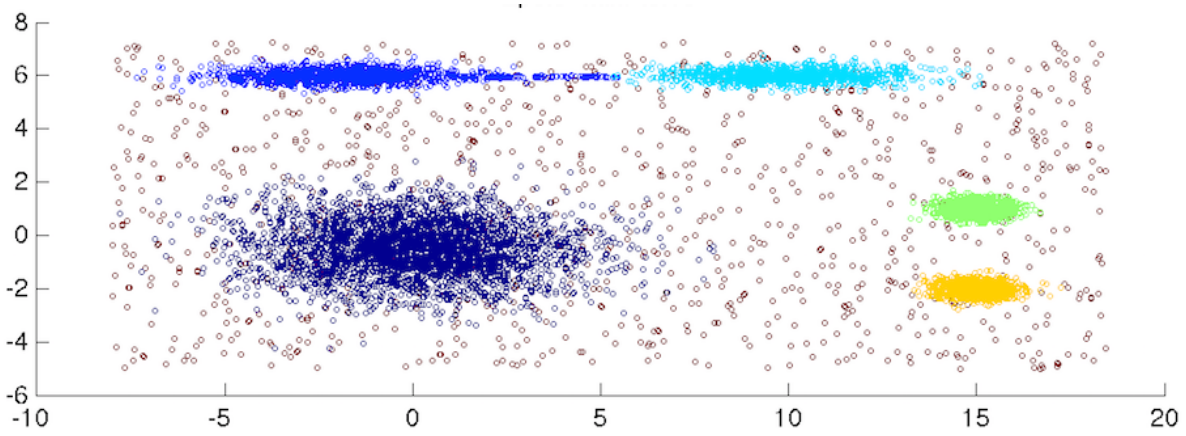

FIGURE 2. Simulated data used in our example proposed initially to study the performance of the CURE algorithm [12].

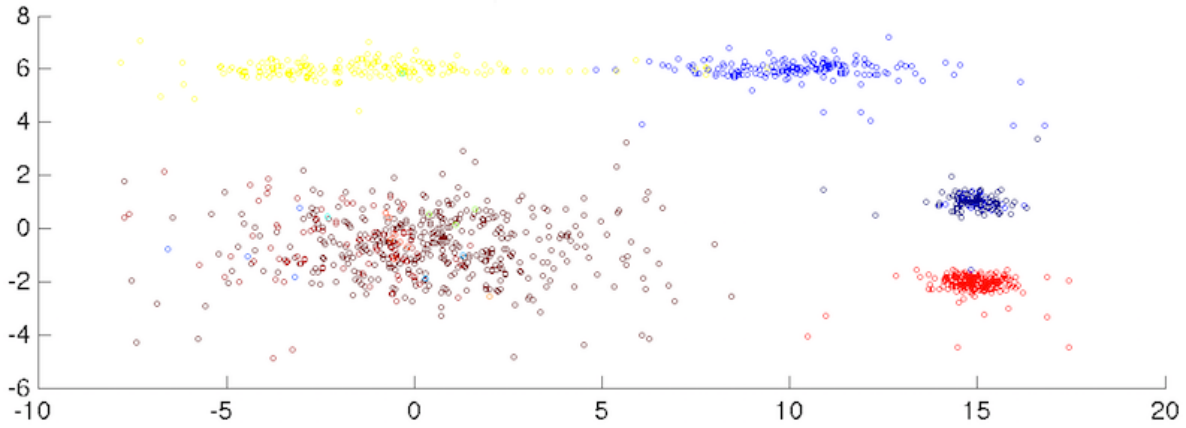

FIGURE 3. Final result after D-SNN clustering with Eps = 50.

represented by the set of clusters L and the set of noise points N is reported by the master node.

ILLUSTRATIVE EXAMPLE

We provide an illustrative example by applying D-SNN over the synthetic 2-dimensional data set shown in Figure 2, which was used by Guha *et al.* [12] when introducing the CURE algorithm. As figure 2 shows, the data set contains five clusters, depicted with different colors. The dataset contains many noise points (more than 2k points over a data set of 70k data points). The algorithm was executed over eight disjoint data partitions created using uniform sampling at random. The SNN similarity was computed using Euclidean neighborhoods of 90 data points. Parameter MinPts is set to 30 and parameter ω is set to 0.3. The attained results when using parameter Eps equal to 50 and 60 are shown in figures 3 and 4, respectively.

Figures 3 and 4 show the effectiveness of our algorithm and its robustness to the presence of noise. By contrasting Figure 2 against 3 and 4, it is noticed that the noise layer was successfully removed by our clustering algorithm in both cases. Note that the Eps value had an impact on the sensitivity of the algorithm to detect noise points, i.e., a higher value for this parameter is accompanied by an increasing rate of points marked as noise as it is especially noticed in the bridge of points that connect both ellipsoids in Figure 2.

C. COST OF THE ALGORITHM

Assuming that D has n data points and the system works over P nodes, the cost of the SNN space construction per node is $\Theta\left(\frac{n^2}{P^2}\right)$. The cost of the labeling process per node is linearly upper bounded by the partition size $\mathcal{O}(\frac{n}{P})$ (the worst case happens when the entire partition is marked as core point). The cost of data transmission per node is ruled by η and is upper bounded by the partition size $\mathcal{O}(\eta \cdot \frac{n}{P})$. In the master node, the size of the centralized sample S is upper bounded by $\mathcal{O}(\eta \cdot n)$. Then, the construction of the SNN space over S costs $\Theta\left(\eta^2 \cdot n^2\right)$. The cost of the labeling process is upper bounded by the size of S. Then its cost is $\mathcal{O}(\eta \cdot n)$.

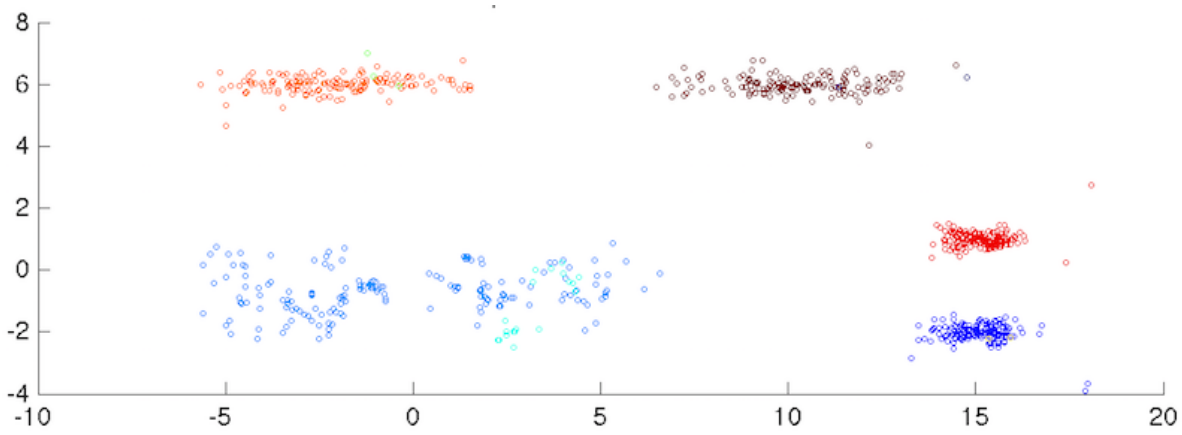

FIGURE 4. Final result after D-SNN clustering with Eps = 60.

TABLE 1. Description of the data sets employed in terms of the number of instances, size of the vocabulary (that is the dimensionality of the document vectors), number of classes and percentage of non-zero values in the document-term matrix.

Dataset	Description	instances	$\|\mathcal{V}\|$	classes	%NNZ
20NG	20-newsgroup data, m5 partition.	4743	41223	5	0.167%
DOE	Department of Energy Abstracts	1664	15755	14	0.365%
FR	Federal Register notes	926	50427	14	1.104%
SJMN	San Jose Mercury News	908	23616	16	0.738%
ZF	Computer select disks by Ziff-Davis	3263	58398	25	0.360%

As the costs are governed by the quadratic costs involved in SNN construction, the total cost of D-SNN is ruled by $\Theta\left(\left(\frac{1}{P^2}+\eta^2\right)\cdot n^2\right)$. The factor $\frac{1}{P^2}+\eta^2$ represents the fraction of C-SNN that D-SNN is able to reduce.

IV. METHODOLOGY AND EXPERIMENTAL RESULTS

In this section, we assess the performance of D-SNN on five data sets comparing its effectiveness against C-SNN [10]. Also, we include two methods in the evaluation: Firstly, an SNN-graph bisection method successfully employed in text clustering tasks [34]. Secondly, a parallel version of K-means++ proposed by Bahmani *et al.* [2] and successfully implemented into the Spark framework. In this section, the former algorithm is denoted as *Graph Clust* and the latter is denoted as *Kmeans||*.

Each data set was partitioned into four disjoint partitions produced by a uniform sampling process at random. Then, each partition was distributed onto different data nodes in which the initial stage procedure was conducted. Then, the results obtained after performing the final stage procedure in the master node were assessed.

A. DATA SETS

The experiments were performed over 5 data sets containing documents represented as vectors in high dimensional term spaces. We use the well-known text data set 20-Newsgroups (M5 partition), a document collection that contains almost 5000 news coming from different subjects, namely computers, motorcycles, baseball, science, and politics. The remaining four document sets were extracted from the Tipster collection.[2]

In specific, we include in our evaluation *DOE*, a data set which contains short abstracts of the Department of Energy, *FR* which contains reports of actions taken by U.S. government agencies, *SJMN*, a collection of news published by the San Jose Mercury News, and *ZF*, a collection of news about computers published by Ziff-Davis Publishing Co.

In table 1, we summarize the number of documents, the size of the vocabulary, the number of classes and the percentage of non zero values (a measure of the sparsity of the collection) per data set. Note that all these collections are quite sparse and also their document vectors are spread onto high dimensional term spaces with tens of thousands of features.

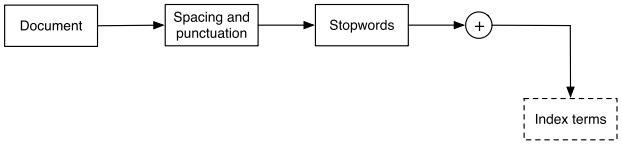

FIGURE 5. Document processing steps used in our work.

B. TEXT PROCESSING

To obtain a representative set of terms for each collection, the documents were processed following the steps indicated in Figure 5. This figure shows standard text processing steps

[2]https://tac.nist.gov/data/data_desc.html

commonly used in information retrieval and text mining tasks. Firstly, as documents within the Tipster collections come in SGML format, their content was retrieved. Then, spacing and punctuation marks were removed. After this step, the set of terms was filtered using an English stop-word list, which aids to remove meaningless words (e.g. *the*, *of*, *by*).

None other process was applied to the documents in this work (e.g., stemming or lemmatizing) in order to keep the text content as similar as possible to the source and especially to allow subsequent generalizations of the performance results without any tie to a specific data set. Finally, the selected terms within each collection were sorted, and then the vocabulary was determined (the vocabulary size reported in table 1 corresponds to the vocabulary obtained after document processing).

After the vocabulary of each collection was built, the number of occurrences of its terms in each document was computed and used to build the vector representation of each document. Let $f_{i,j}$ be the number of occurrences of a term i of the vocabulary $\mathcal{V}$ in the document j of the collection. Let n_i be the number of documents of the collections that contain the term i. The following expression quantifies the weight of term i within a document j:

$$w_{i,j} = \frac{f_{i,j}}{\max_{t \in \mathcal{V}} f_{t,j}} \cdot \log \frac{n}{n_i}$$

which corresponds to the Tf-Idf weighting scheme of the term i for the document j.

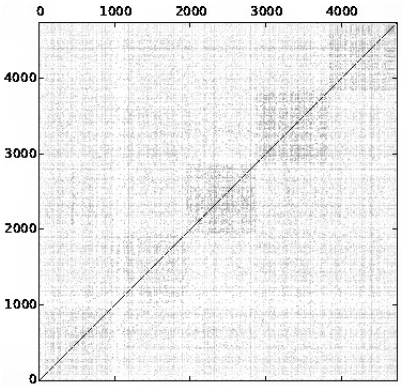

FIGURE 6. Similarity matrix plot for the 20 Newsgroup dataset (20NG).

We use the cosine proximity to measure the similarity between each pair of documents in each data set. To understand differences and resemblances across data sets, we plot the similarity matrix of each dataset by sorting the documents consecutively by their class labels. A gray-scale was employed for each plot in which darker spots denote higher similarity values. These matrices are depicted in figures 6 to 10.

The matrices mentioned above show darker rectangular patches along the diagonal line, showing the presence of clusters. Also, 20NG (Fig. 6) and ZF (Fig. 10) show pairwise

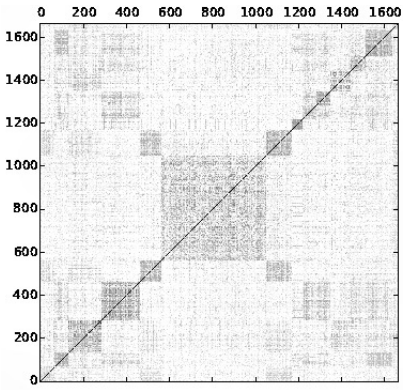

FIGURE 7. Similarity matrix plot for the Department of Energy dataset (DOE).

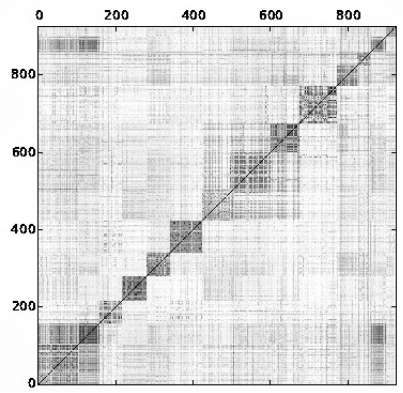

FIGURE 8. Similarity matrix plot for the Federal Register dataset (FR).

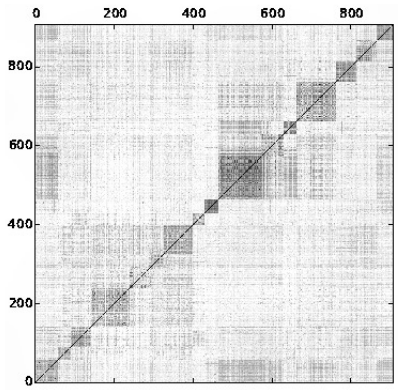

FIGURE 9. Similarity matrix plot for the San Jose Mercury News dataset (SJMN).

similarities not much higher than the ones outside the clusters, suggesting that the data points in these data sets are very sparse. In the case of the DOE collection (Fig. 7), some spots outside the diagonal appear suggesting the existence of cluster overlapping. Finally, both SJMN (Fig. 9) and FR (Fig. 8) collections show small squares embedded into big squares along the diagonal, suggesting the existence of nested clusters.

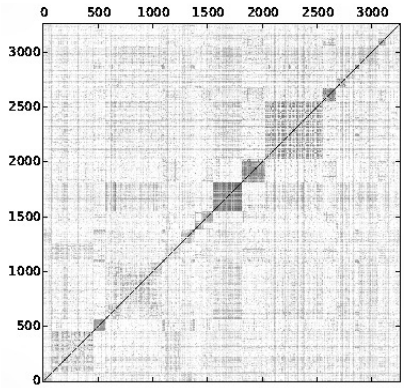

FIGURE 10. Similarity matrix plot for the Ziff/Davis dataset (ZF).

C. CLUSTERING QUALITY MEASURES

In this work, we used external validation measures to assess cluster quality. To this end, we employed the **Adjusted-Mutual-Information**, **Homogeneity**, **Completeness** and **V-Measure**.

The **Adjusted-Mutual-Information** (AMI) measure was proposed by Nguyen *et al.* [29] and evaluates how much information of a ground truth class is represented into the resulting clustering. The **Homogeneity** (HOM) metric measures the extent in which the clusters contain only data points which are members of a single class. The **Completeness** (COM) metric accounts for the extent to which all the data points that are members of a given class belong to the same cluster. The **V-Measure** (VM) corresponds to the harmonic mean between **Homogeneity** and **Completeness**. These metrics take positive values within [0, 1], except for the Adjusted-Rand-Index, which is within $[-1, 1]$. For all of these measures, larger values denote a better clustering quality.

D. GRID SEARCH OF PARAMETERS

To report the best performance found for each algorithm, we conducted a grid search for K, Eps and MinPts. The results were evaluated in terms of **V-Measure**. We used the same grid for C-SNN and D-SNN. The grid used consists of 432 configurations with K in {15, 30, 50, 70, 90, 110}, Eps in {3, 5, 8, 10, 15, 20, 25, 30, 35, 40, 45, 50} and MinPts in {5, 10, 15, 20, 25, 30}. For the *Graph Clust* algorithm we tested the same values for K but no difference in the performance was detected. Additionally, since the *Graph Clust* and *Kmeans||* algorithms require the number of groups as a parameter (k-direct algorithms), this parameter was set to the known number of classes of each dataset, allowing to evaluate a fair comparison between our algorithm and our competitors. We show the best parameters for C-SNN and D-SNN in Tables 2 and 3, respectively.

In the case of D-SNN, we run our experiments using $w = 0.3$ over 4 partitions.

E. RESULTS

The results of our experiments are shown in Table 4. Bold fonts indicate the best result for each experimental configuration.

TABLE 2. Parameters of the C-SNN clustering algorithm selected after the grid search of parameters.

Dataset	K	Eps	MinPts
20NG	110	25	30
DOE	70	25	30
FR	50	25	20
SJMN	50	20	30
ZF	90	40	25

TABLE 3. Parameters of the D-SNN clustering algorithm selected after the grid search of parameters.

Dataset	K	Eps	MinPts
20NG	30	8	25
DOE	30	10	20
FR	30	10	5
SJMN	30	10	10
ZF	30	10	15

TABLE 4. Performance attained by the graph clustering algorithm, C-SNN algorithm, and D-SNN over the text collections. The best values for each measure in each dataset appear in bold fonts.

	Dataset	AMI	HOM	COM	VM
Graph Clust	20NG	0.6421	0.6619	0.6425	0.6521
	DOE	**0.7030**	0.7095	**0.7461**	0.7273
	FR	0.7266	0.7375	0.7452	0.7413
	SJMN	0.7367	0.7505	0.7657	0.7580
	ZF	0.5444	0.5593	0.6015	0.5796
C-SNN	20NG	0.3953	0.3990	0.4793	0.4355
	DOE	0.6370	0.6476	0.6711	0.6591
	FR	**0.7834**	0.7969	0.7919	0.7944
	SJMN	0.6820	0.7732	0.6960	0.7326
	ZF	0.5084	0.5750	0.5238	0.5482
Kmeans||	20NG	0.4696	0.4702	0.5607	0.5115
	DOE	0.1255	0.1344	0.6767	0.2242
	FR	0.3236	0.3426	0.7062	0.4614
	SJMN	0.2299	0.2518	0.6767	0.3670
	ZF	0.0536	0.0648	0.6072	0.1171
D-SNN	20NG	**0.8218**	**0.8262**	**0.9167**	**0.8691**
	DOE	0.7029	**0.8794**	0.7227	**0.7934**
	FR	0.7546	**0.8947**	**0.7784**	**0.8325**
	SJMN	**0.7836**	**0.8052**	**0.8040**	**0.8046**
	ZF	**0.7701**	**0.9882**	**0.7877**	**0.8766**

Table 4 shows that D-SNN obtains the best performance in almost all the experiments, showing that our algorithm can improve clustering quality by conducting density sampling over disjoint partitions. These results are due to the robustness of D-SNN to the presence of noise points. Also, as density sampling is conducted over disjoint partitions, the performance of the algorithm increases. Then, the robustness of the algorithm to the presence of noise limits the influence of noise points on clustering solution. The last finding explains why D-SNN outperforms C-SNN.

As depicted in figures 6 and 10, 20NG and ZF exhibit a low discrimination power of the underlying similarity measure in the vector term space spanned by the document vectors in each collection. Table 4 shows that this phenomenon affects both centralized algorithms, especially C-SNN in 20NG, by hindering their capability of identifying homogeneous clusters. The lower values of the **AMI** scores support this finding.

Also, the similarity matrix for DOE (Fig. 7) shows an unbalanced data set, a fact that can be evidenced by observing the uneven patch sizes along the diagonal in its similarity matrix. Table 4 shows that D-SNN works well under this scenario. This finding suggests that the shared near neighbor approach helps to address this obstacle.

Finally, similarity matrices for data sets FR (Fig. 8) and SJMN (Fig. 9) show embedded square patches along the diagonal, which suggests a hierarchical structure of the underlying classes. Table 4 shows that D-SNN works well under this scenario, and accordingly, the impact on the performance of nested clusters is successfully drawn by D-SNN.

In summary, our experiments show that the extraction of core-points, together with the sampling strategy followed to transmit the representatives to the central node, enable more precise discrimination of each group. Regarding the *Kmeans||* algorithm implemented within the Spark framework, it attained poor performance scores over all data sets. This observation is due to the high dimensional nature of text data. Another factor that explains this poor performance is class unbalance. Finally, it is important to stress that Spark *Kmeans||* currently corresponds to one of the most successful implementations of a distributed clustering algorithm in terms of scalability.

F. AN HEURISTIC FOR PARAMETER TUNING

In the previous section, we showed that D-SNN outperforms C-SNN in almost all the experiments. However, the comparison was made using the best values found for the parameters using a grid search. The performance of each parameter assignment was evaluated in terms of **V-Measure**, a performance measure that uses actual labels to measure cluster quality. In an unsupervised learning scenario, D-SNN will work with unlabeled data. As observed by Rukmi *et al.* [38], SNN is an algorithm sensitive to the choice of the parameters k, MinPts, and Eps. Indeed, a high value of k determines that the number of neighbors of each document may include documents with low similarity, deteriorating the quality of the clusters found. Likewise, a high value of Eps produces more documents classified as noise. Finally, a high value of MinPts causes SNN underestimating the actual value of the number of clusters. Given the implications that an unfortunate choice of SNN parameters has on the quality of the results, a data-driven tuning strategy of these parameters is required.

In this section, we discuss how to tune the parameters of D-SNN on unlabeled data. In order to do this, we present a useful and straightforward data-driven heuristic to tune Eps and MinPts.

Our heuristic is inspired by the sorted k-sim graph used by Ester *et al.* [9] in DBSCAN. The k-sim graph corresponds to the graph of similarities of each point to its k-nearest neighbor. The sorted k-sim graph is the k-sim graph with similarities sorted in decreasing order. As was pointed out by Ester *et al.* if we use k = MinPts, the sorted k-sim graph shows an inflection point. They selected Eps from the sorted k-sim graph, a value that is identified using the knee of the curve. However, they did not provide a way to determine the value of MinPts.

Our heuristic comes from an observation made on sorted k-sim graphs. Note that, in addition to Eps and MinPts, C-SNN and D-SNN has an additional parameter named k, which determines the size of the neighborhood where the SNN space is computed. We calculate a set of sorted k-sim graphs for different values of k. Note that k corresponds to the parameter used in the SNN space for similarity calculation, and k is the parameter of the sorted k-sim graph, then k and k are different.

We noted that the shape of each plot is almost insensitive to k. Different values of k produce a rescaling of the plot in terms of distances, but the shape remains the same. As sorted k-sim graphs are insensitive to k, we need to fix k considering other criteria. On the one hand, an important criterion to fix k is to avoid unnecessary computation. A high value of k will demand the maintenance of large neighborhoods lists for SNN calculation. On the other hand, a low value of k may discard close points in high-density regions. Then, we propose to use a value for k, considering the density of the dataset. This parameter will be proportional to the density of the data set in the original space. The higher the density of the space, the higher the value of k. To avoid a linear increment on k due to data size, we introduce a sublinear factor $\log(n_p)$, where n_p corresponds to the size of the data partition. Note that for C-SNN, $n_p = n$, where n corresponds to the size of the dataset. Therefore, we estimate k by:

$$k = \frac{n \cdot \mathsf{nnz}}{100} \cdot \log(n_p).$$

For instance, in 20NG as $n = 4743$, and $\mathsf{nnz} = 0.167$, then $k \approx 63$ for C-SNN and 52 for D-SNN. Then we round k to the closest number multiple of 5.

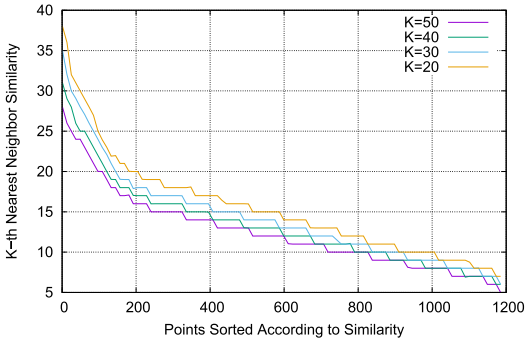

FIGURE 11. k-sim graph for 20 Newsgroup dataset (20NG).

To tune MinPts, we use sorted k-sim plots for several values of k. The highest value of k = k (rounded), is useful to start the search of k from k in decreasing order in decrements of 10. These curves are shown in figures 11 to 15. Note that low values of k produce less convex curves and high values of similarities. Ester *et al.* [9] observed convex curves for k-sim graphs (DBSCAN was evaluated using k=4). We propose to

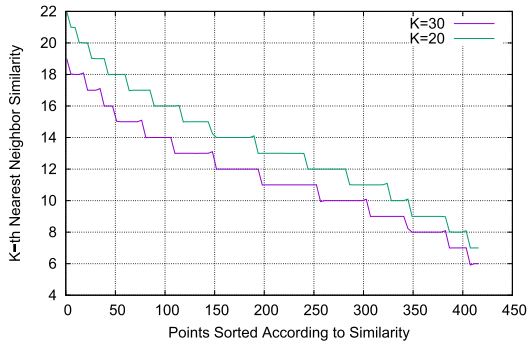

FIGURE 12. k-sim graph for the Department of Energy dataset dataset (DOE).

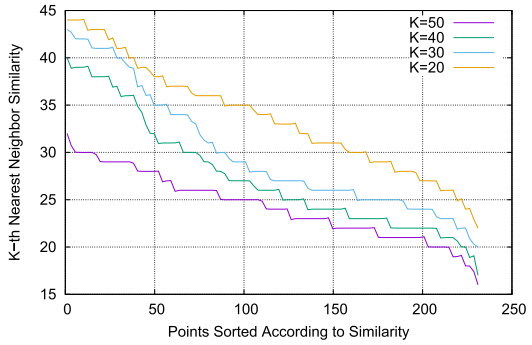

FIGURE 13. k-sim graph for the Federal Register dataset (FR).

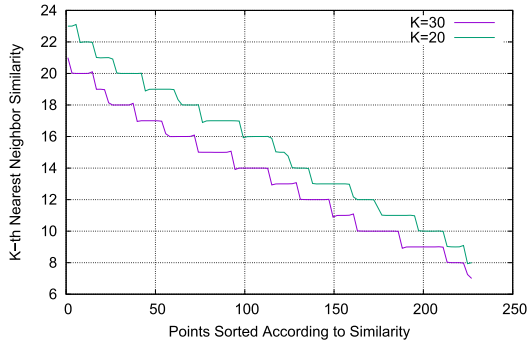

FIGURE 14. k-sim graph for the San Jose Mercury News dataset (SJMN).

use the sensitivity to k of sorted k-sim graphs to tune MinPts and Eps.

We set the value of MinPts to the lower value of k that produces convexity in the k-sim plot. We conduct a search for MinPts, starting from k. A binary search on k is advisable to avoid unnecessary computation of k-sim curves when k is high. Figures 11 to 15 show the results of this procedure over the five data sets used in this work. In some data sets, the curves are almost equivalent in terms of convexity (e.g., see figures 11 and 14). In these cases, we choose the value that is in the middle of the search range. In other cases, the change in convexity is more notorious (e.g., see figures 13 and 15). Note that Eps is retrieved using the knee of the curve (the projection of the point to the y-axis). These figures also show that the values found using our heuristic

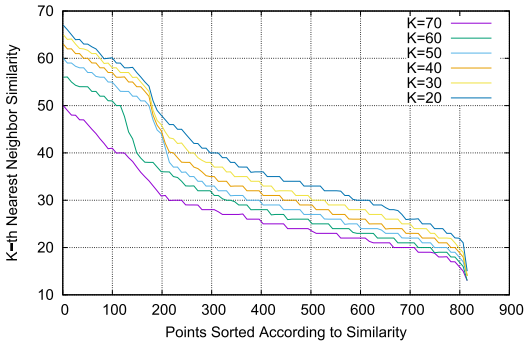

FIGURE 15. k-sim graph for the Ziff/Davis dataset (ZF).

are similar to the ones showed using grid-search. The only exception to this finding was observed in 20NG, where the value of k was over-estimated. The results of both algorithms using our tuning heuristic in terms of cluster quality are shown in Table 5.

TABLE 5. Performance attained by C-SNN and D-SNN over the text collections using our tuning heuristic. Clustering quality results in terms of V-Measure are shown in the last column.

Data Set	Algorithm	k	MinPts	Eps	VM
20NG	C-SNN	65	30	20	0.36
DOE	C-SNN	40	30	20	0.53
FR	C-SNN	70	40	50	0.61
SJMN	C-SNN	45	30	25	0.65
ZF	C-SNN	95	60	60	0.44
20NG	D-SNN	50	30	15	0.65
DOE	D-SNN	35	20	10	0.79
FR	D-SNN	35	20	25	0.69
SJMN	D-SNN	35	30	15	0.67
ZF	D-SNN	55	30	40	0.74

Table 5 shows the results in terms of the **V-Measure** after using our tuning procedure. This scenario is close to a real *in production* scenario, where the clustering task is conducted over unlabeled data, and accordingly, the search for the optimum clustering solution is, at some extent blind. We note that D-SNN outperforms C-SNN in all the comparisons. The results show that C-SNN and D-SNN are sensitive to parameter tuning. Accordingly, an incorrect parameter setting may decrease the performance of D-SNN.

D-SNN performs well in these experiments. We understand that the use of partitions and the distributed nature of the algorithm can reduce the impact of parameter tuning on performance. In general, density-based clustering algorithms are susceptible to parameter tuning. We evidence that a distributed density-based algorithm can limit parameter sensitiveness, finding an additional suitable property to our algorithm.

In summary, the results show that our algorithm performs well in distributed collections being precise in detecting the cluster structure of the analyzed data. In addition, the data-driven tuning method reduces the impact of parameter sensibility.

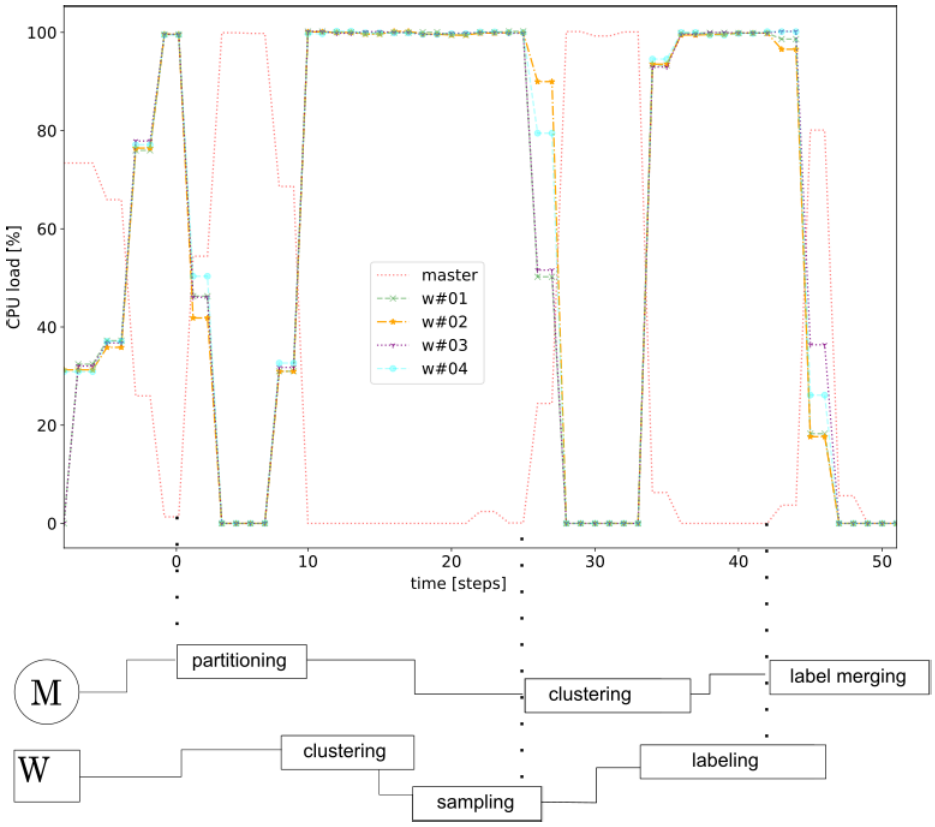

FIGURE 16. CPU Load of a single DSNN execution with 4 workers plus the Master node.

G. EMPIRICAL ASSESSMENT OF THE COMPUTATIONAL COST

In order to illustrate the distributed nature of the proposal in a real scenario, a randomly selected 50% of the full *20NG* dataset (10k documents) is clustered by DSNN with 4 workers. The CPU load of each node is recorded and contrasted against the stages of the algorithm described in section III-B. Figure 16 depicts this information along with two task flows corresponding to the main stages carried by the master and each one of the workers.

In the first part of the plot of Figure 16, until time step 0, the master and the workers records a computational load due to initialization operations. From time step 0 onward, data partitioning and distribution are performed by the master until time step 10. Some steps before time step 10, *Stage 1* begins by invoking the clustering and sampling tasks in each worker. Around time step 25, all workers start to send back their core points along with the sampled points to the master, which starts a centralized clustering task over these summary data. After this procedure is finished around the time step 32, labels are sent from the master to each worker to re-cluster its initially assigned partition and send back the updated labeling to the master node. Finally, around time step 42, the master starts to consolidate the final clustering of the whole dataset. The plot illustrates how the algorithm works in a distributed environment and shows the alternation in the computational load between the master and the worker nodes. It can be seen that the workers carry a good part of the load and that the stages executed by the master have a shorter duration, which favors the scalability of the algorithm preventing the master from transforming into a bottleneck for DSNN.

H. LIMITATIONS OF THE METHOD

Undoubtedly, one of the limitations of clustering algorithms is their sensitivity to hyperparameters. In the case of density-based algorithms, such as DBSCAN and our distributed version D-SNN, the parametric sensitivity determines the quality of the clusters found. By providing a data-oriented parameter tuning strategy, D-SNN decreases the impact of this weakness.

Clustering algorithms tend to be noise sensitive. Since D-SNN uses density-based sampling, it is capable of dealing with noise that produces points in low-density regions. A limitation of density-based clustering algorithms is the inability to distinguish between data sources and noisy point sources. Consequently, D-SNN could confuse a point source of noise, marking it as a cluster if the density sampling conditions allow choosing enough samples from such a source.

V. CONCLUSION AND FUTURE WORK

In this work, a distributed clustering method able to deal with text collections was proposed. In order to assess its

utility, especially for recovering the cluster structure underlying each collection, its performance was compared against two centralized approaches (C-SNN and Graph Clust). Also, its performance was compared to k-means||, a fast implementation of the famous prototype-based clustering algorithm implemented in Spark. Our algorithm outperformed all these algorithms on five different data sets. The results corroborate the ability that D-SNN has to work in a high-dimensional dataset, such as those produced with tf-idf vectorization on text.

Five real text collections were employed to show the weaknesses and strengths of the shared-nearest-neighbor approach. The results indicate that D-SNN, in addition to its power to process distributed data collections, maintains the functional properties of C-SNN. Also, D-SNN improves the capability of dealing with high-density spaces.

The proposed method can be easily extended to impute a label for each data point of a given collection. That is, after the final stage in the central node, the labels of the clustered core-points can be re-transmitted to their original workers. Then, each document can be labeled according to their nearest core-point. Nevertheless, although this task is essential for automatic tagging (e.g., automatic generation of Web directories), we think that the main contribution of this work lies in collection summarization. Our algorithm can obtain a summary of the groups hidden in an extensive and distributed collection, a task that is unfeasible on a single machine.

As future work, we propose to provide an implementation of D-SNN on a big-data framework as Mahout or Spark. With such an implementation, we will able to take advantages from the scalability of our proposal. As our experiments show that D-SNN is very useful in terms of clustering quality, the addition of scalability properties will provide at the same time a fast and effective distributed clustering algorithm for researchers and practitioners.

REFERENCES

[1] R. Azimi and H. Sajedi, "A decentralized gossip based approach for data clustering in peer-to-peer networks," *J. Parallel Distrib. Comput.*, vol. 119, pp. 64–80, Sep. 2018.
[2] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proc. VLDB Endowment*, vol. 5, pp. 622–633, Mar. 2012.
[3] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed κ-means and κ-median clustering on general topologies," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 26, 2013, pp. 1–9.
[4] F. Crestani and I. Markov, "Distributed information retrieval and applications," in *Proc. 35th Eur. Conf. Inf. Retr.* Berlin, Germany: Springer, 2013, pp. 865–868.
[5] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: Scalable online collaborative filtering," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 271–280.
[6] C. M. De Vries, L. De Vine, S. Geva, and R. Nayak, "Parallel streaming signature EM-tree: A clustering algorithm for Web scale applications," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 216–226.
[7] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-Scale Parallel Data Mining*, vol. 1759, no. 802. Berlin, Germany: Springer, 1999, pp. 245–260.
[8] A. Ene, S. Im, and B. Moseley, "Fast clustering using MapReduce," in *Proc. KDD*, 2011, pp. 681–689.
[9] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.
[10] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proc. SIAM Int. Conf. Data Mining*. Philadelphia, PA, USA: SIAM, 2003, pp. 47–58.
[11] G. Forman and B. Zhang, "Distributed data clustering can be efficient and exact," *ACM SIGKDD Explor. Newslett.*, vol. 2, no. 2, pp. 34–38, 2000.
[12] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," *ACM SIGMOD Rec.*, vol. 27, no. 2, pp. 73–84, 1998.
[13] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 593–599.
[14] E. Januzaj, H.-P. Kriegel, and M. Pfeifle, "Scalable density-based distributed clustering," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discovery*. Berlin, Germany: Springer, 2004, pp. 231–244.
[15] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Trans. Comput.*, vol. C-22, no. 11, pp. 1025–1034, Nov. 1973.
[16] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson, "Distributed clustering using collective principal component analysis," *Knowl. Inf. Syst.*, vol. 3, no. 4, pp. 422–448, 2001.
[17] M. Klusch, S. Lodi, and G. Moro, "Distributed clustering based on sampling local density estimates," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2003, pp. 485–490.
[18] H.-P. Kriegel, P. Kr, A. Pryakhin, and M. Schubert, "Effective and efficient distributed model-based clustering," in *Proc. 5th IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2005, p. 8.
[19] T. Li, S. Zhu, and M. Ogihara, "Algorithms for clustering high dimensional and distributed data," *Intell. Data Anal.*, vol. 7, pp. 305–326, Feb. 2003.
[20] Y. Liang, M.-F. Balcan, and V. Kanchanapally, "Distributed PCA and k-means clustering," in *Proc. Big Learn. Workshop NIPS*, 2013, pp. 1–8.
[21] J. Liu, J. Z. Huang, J. Luo, and L. Xiong, "Privacy preserving distributed DBSCAN clustering," in *Proc. Joint EDBT/ICDT Workshops*, 2012, pp. 177–185.
[22] H. Mashayekhi, J. Habibi, T. Khalafbeigi, S. Voulgaris, and M. Van Steen, "GDCluster: A general decentralized clustering algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1892–1905, Jul. 2015.
[23] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*. New York, NY, USA: ACM, 1996, pp. 103–114.
[24] M. Mendoza, M. Marín, V. Gil-Costa, and F. Ferrarotti, "Reducing hardware hit by queries in Web search engines," *Inf. Process. Manage.*, vol. 52, no. 6, pp. 1031–1052, 2016.
[25] S. Merugu and J. Ghosh, "Privacy-preserving distributed clustering using generative models," in *Proc. 3rd IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2003, pp. 211–218.
[26] N. K. Nagwani, "Summarizing large text collection using topic modeling and clustering based on MapReduce framework," *J. Big Data*, vol. 2, p. 6, Jun. 2015.
[27] M. Sarnovsky and N. Carnoka, "Distributed algorithm for text documents clustering based on k-means approach," in *Advances in Intelligent Systems and Computing*, vol. 430. Cham, Switzerland: Springer, 2016, pp. 165–174.
[28] D. Talia, "Parallelism in knowledge discovery techniques," in *Proc. 6th Int. Conf. Appl. Parallel Comput. Adv. Sci. Comput.* Berlin, Germany: Springer, 2002, pp. 127–138.
[29] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Oct. 2010.
[30] N. K. Visalakshi, K. Thangavel, and P. Alagambigai, "Distributed clustering for data sources with diverse schema," in *Proc. 3rd Int. Conf. Converg. Hybrid Inf. Technol.*, 2008, pp. 1056–1061.
[31] X. Xu, J. Jäger, and H. Kriegel, "A fast parallel clustering algorithm for large spatial databases," in *High Performance Data Mining*, vol. 290. Springer, 1999, pp. 263–290.
[32] J. Yi, L. Zhang, J. Wang, R. Jin, and A. K. Jain, "A single-pass algorithm for efficiently recovering sparse cluster centers of high-dimensional data," in *Proc. 31st Int. Conf. Mach. Learn.*, vol. 3, Beijing, China, 2014, pp. 2112–2127.
[33] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao, "A parallel clustering algorithm with MPI—MKmeans," *J. Comput.*, vol. 8, no. 1, pp. 10–18, 2013.

[34] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in *Proc. 11th Int. Conf. Inf. Knowl. Manage.*, 2002, pp. 515–524.
[35] C. Haydar and A. Boyer, "A new statistical density clustering algorithm based on mutual vote and subjective logic applied to recommender systems," in *Proc. 25th Conf. User Modeling, Adaptation Personalization (UMAP)*, 2017, pp. 59–66.
[36] M. Ravichandran, K. M. Subramanian, P. Ganesan, and R. Jothikumar, "A modified method for high dimensional data clustering based on the combined approach of shared nearest neighbor clustering and unscented transform," *J. Comput. Theor. Nanosci.*, vol. 15, nos. 6–7, pp. 2050–2054, 2018.
[37] S. Kumari, S. Maurya, P. Goyal, S. S. Balasubramaniam, and N. Goyal, "Scalable parallel algorithms for shared nearest neighbor clustering," in *Proc. IEEE 23rd Int. Conf. High Perform. Comput. (HiPC)*, Dec. 2016, pp. 72–81.
[38] A. M. Rukmi, D. B. Utomo, and N. I. Sholikhah, "Study of parameters of the nearest neighbour shared algorithm on clustering documents," *J. Phys., Conf. Ser.*, vol. 974, no. 1, pp. 12–61, 2018.
[39] C. J. Plattel, "Distributed and incremental clustering using shared nearest neighbours," M.S. thesis, Dept. Inf. Comput. Sci., Utrecht Univ., Utrecht, The Netherlands, 2014.
[40] J. Zamora, H. Allende-Cid, and M. Mendoza, "A distributed shared nearest neighbors clustering algorithm," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (Lecture Notes in Computer Science), vol. 10657, M. Mendoza and S. Velastín, Eds. Cham, Switzerland: Springer, 2018.

JUAN ZAMORA received the Ph.D. degree from the Universidad Técnica Federico Santa María, Chile, in 2016. He is currently an Assistant Professor with the Instituto de Estadística of Pontificia Universidad Católica de Valparaíso. His research interests include data mining, text mining, and clustering algorithms.

HÉCTOR ALLENDE-CID received the Ph.D. degree from the Universidad Técnica Federico Santa María, Chile, in 2015. He is currently an Assistant Professor with the Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso. He is also the Founder and the current President of the Chilean Association of Pattern Recognition. His research interests include supervised algorithms, distributed regression methods, and image processing.

MARCELO MENDOZA received the master's degree in informatics from the Universidad Técnica Federico Santa María, Chile, and the Ph.D. degree in computer science from the Universidad de Chile. He held a postdoctoral position at Yahoo Research. He is currently a Faculty Professor with the Department of Informatics, Universidad Técnica Federico Santa María, where he is also the Head of the master's in informatics program, and an Electronic Engineer. He is the Founder and the former President of the Chilean Association of Pattern Recognition. He is currently a Researcher with the Valparaíso Center of Science and Technology and also an Associate Researcher with the Millennium Institute for Foundational Research on Data. His research interests include text mining, information retrieval, and data mining in social networks.