PROPOSAL ABSTRACT:

The maximum length of this file is **10 pages** (Must use letter size, Verdana size 10 or similar)**.** For an adequate evaluation of your proposal merits,this file must include the following aspects: Proposal description, Bibliographical References, Hypothesis, Goals, Methodology, Work Plan, Work in progress (if applicable) and Available Resources. Make sure to describe the relevance of the proposal topic in relation to the state of the art in the field. Keep in mind the Bases del Concurso FONDECYT de Postdoctorado 2018 and Application Instructions.

1 Introduction

As a consequence of the explosive growth of the WEB, the integration of search engines such as Google into personal computers and mobile devices and the wide use of social networks, the task of clustering text data has gained importance due to the necessity of unraveling the underlying categories in large volumes of text data (e.g. Tweets or WEB pages). Nowadays the generation of large amounts of documents surpasses the computational power of personal computers and even of high performance computers. As an example, it is estimated that the amount of WEB pages that popular search engines such as Yahoo! and Google index is higher than 40 billion of pages[1]. It is therefore of great interest to develop algorithmic techniques capable of automatically organize, classify and summarize document collections distributed in multiple machines of a network, and that also perform efficiently in modern hardware, particularly within parallel processing frameworks in multi-core architectures. In real problems such as *Collection Selection* for distributed document databases, in which for a given query the computer node containing the most suitable sub-collection must be selected to answer it, the challenges related to the scalability and efficiency of *Knowledge Discovery* methods have become very important. Traditional algorithms often assume that the whole dataset is loaded into main memory (RAM) and thus every document can be accessed at any time with no access latency. Actual real problems present though challenges since the sizes of the collections explored nowadays are much bigger than the amount of available RAM in modern computers[2], either because of the number of documents or the size of the computational representation of the documents.

Often text data is arranged into digital collections of documents containing a variable number of words. After extracting the terms appearing in every document an overall vocabulary that summarizes the words contained in the collection is built. In real text collections, this vocabulary may contain from tens of thousands to several hundred of thousands of words. In order to capture the lexical abundance of a language each document is computationally treated as a bag of words contained in the vocabulary, which in turn is represented as a vector whose dimensionality equals the size of the vocabulary. This enormous size of the input data is far bigger than the size that traditional clustering algorithms can handle. Because of this, the task of automatic document clustering involves high computational and storage (RAM and secondary memory) costs. Along with this, when the number of documents is large (tens or hundreds of thousand and even millions of items) traditional techniques for processing and clustering documents are insufficient under the current computational capabilities of a single machine and even high performance machines. Even in the most favorable scenario the storage and computational power tackle this challenge but the response time are excessive.

There are three approaches that have been successfully applied to the construction of clustering algorithms capable of processing large volumes of data. The first one introduces constraints on the number of passes allowed on a document (related to the amount of time it is loaded into main memory). The second one exploits current multi-core architectures to perform parallel processing of the data. Both strategies do not tackle the problem on scenarios in which the data is distributed. The last one combines the computational power of a single machine together with the scalable storage and processing capabilities of a distributed system in which data partitions are independently processed by several machines connected through a network.

Parallel processing architectures

A multi-core processor (CPU) is a single chip that combines two or more independent processing units called *cores*. These processors are typically shared-memory architectures designed to support general purpose operations and contain from 2 to several tens of cores optimized for sequential serial processing of a set of instructions. A Graphical-Processing-Unit (GPU) is a massively parallel processor consisting of hundreds or even thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously. At its beginning, GPUs were employed for hardware-assisted bitmap operations to assist in the display and usability of graphical operating systems. The release of the first GPU built with NVIDIA's CUDA Architecture enabled the usage of this hardware for general purporse computation [Sanders and Kandrot, 2010].

The last hybrid approach previously mentioned seems promising since it allows to exploit the local capabilities of single computers with multi-core architectures without sacrificing scalability to large data volumes because of its distributed design. Within this path there are two contexts regarding the data generation scenario. On the one hand, in some problems where the dataset is large but collected in a centralized fashion, the strategy employed consists in partitioning the collection into

[1] `http://www.worldwidewebsize.com`

[2] Amount of primary memory in modern computers reaches about 128 Gigabytes but a medium size document collection such as RCV1 can reach near 1 Terabyte (depending on the followed text processing procedure).

several machines or nodes of a network. This scheme leads to the transmission of a lot of data during the execution of the algorithm [Nagwani, 2015]. On the other hand, there are some problems where the data is generated in a distributed fashion and it is not feasible to centralize the data because of high transmission costs or privacy issues [Jagannathan et al., 2005; Liu et al., 2012], e.g. search engines work with document collections originated and stored in different geographical locations.

2 Problem Definition

Let $X = \{X_1, X_2, \ldots, X_p\}$ be a collection of documents where $\bigcup_{i=1}^{p} X_i = X$ and $\forall i \neq j \in [1, \ldots, p], X_i \cap X_j = \emptyset$. This collection is partitioned into p different machines of a network, and also each subset X_i consists of n_i vectors in $\mathbb{R}^d$. Additionally, in a distributed environment, data sites may be homogeneous or heterogeneous, depending if different sites contain data for exactly the same set of features or for different set of features respectively. The distributed data clustering task consists in obtaining k data partitions $C_1, C_2 \ldots, C_k$ where $\bigcup_{i=1}^{k} C_i = X$ and also $\forall i \neq j \in [1, \ldots, k], C_i \cap C_j = \emptyset$, such that each one of them represents a topical category of the overall collection and in turn all the categories are covered by the partitions or clusters C_i. Expressed in a different manner, this means that every cluster contains more similar documents according to their content in comparison to the other ones contained in different clusters. The similarity between document vectors is usually measured by means of a function $S : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$, e.g. the cosine similarity function:

$$S(x_a, y_b) = \frac{< x_a, x_b >}{\|x_a\|x_b|},$$

where $< u, v >$ with $u, v \in \mathbb{R}^d$ denotes the inner product between two vectors and $|u|$ corresponds to the Euclidean norm of a vector. In a nutshell, this task consists in finding a structure of compact and homogeneous groups with respect to a similarity value between their members that accounts for their content.

An overall operation mode of the distributed data clustering techniques consists of four stages: Initially, the dataset is partitioned into several nodes. Next, each node generates a clustering model over its corresponding local subset. These local models are then transmitted to a central node (e.g. A model can consists of representative points identified from a local clustering), which integrates the local models in a global solution. Eventually, this global model can be re-transmitted to the other nodes in order to refine their local models and then, by performing the same previous steps, a new depurated global model is built. This scheme is depicted in Figure 1.

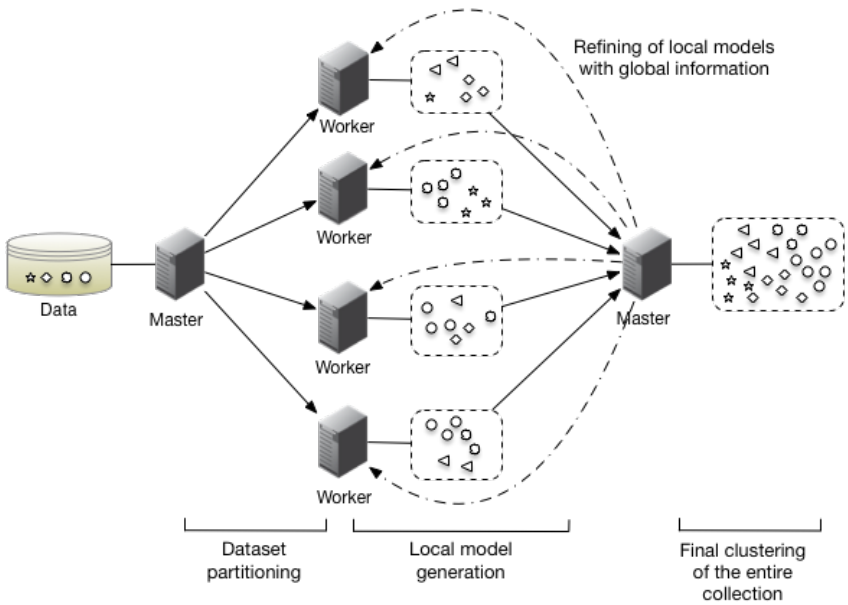

Figure 1: Overall scheme of operation of the distributed data clustering techniques.

3 Parallel and Distributed approaches to High Dimensional Data Clustering

3.1 Main contributions on parallel clustering algorithms

[Xu et al., 2002] presented a parallel version of *DBSCAN* (*PDBSCAN*). The authors presented the shared-nothing architecture with multiple computers interconnected through a network. A fundamental component of a 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. A performance evaluation showed that PDBSCAN offers nearly linear speed-up and has excellent scale-up and size-up behavior.

In [Dhillon and Modha, 1999], the authors 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 approached the optimal as the number of data points increased.

Another scalable approach based on secondary memory consisted in designing algorithms able to work within the MapReduce framework. In this context we highlight the contribution made by [Ene et al., 2011] in which they tackled the K-Median problem by using MapReduce. The authors developed a fast clustering algorithm with constant factor approximation guarantees. The algorithm employed a sampling strategy to decrease the data size and after that a time consuming clustering algorithm such as local search or Lloyd's algorithm was performed on the resulting dataset.

Another parallel approach for K-Means was presented by [Bahmani et al., 2012] and it is called K-Means++. The initialization of the K-Means algorithm is crucial, and the authors claimed that the proposed algorithm obtained an initial set of centers that is probably close to the optimum solution. A major downside of K-Means++ is its inherent sequential nature, which limits its applicability to massive data: one must make K passes over the data to find a good initial set of centers. In this work the authors showed how to drastically reduce the number of passes needed to obtain, in parallel, a good initialization. This is unlike prevailing efforts on parallelizing K-Means that have mostly focused on the post-initialization phases of K-Means. The proposed initialization obtained a nearly optimal solution after a logarithmic number of passes, and then showed that in practice a constant number of passes suffices.

3.2 Clustering after applying dimensionality reduction

[Kargupta et al., 2001] proposed a method to obtain the Principal Components (PCA) over heterogeneous and distributed data. Based on this contribution on dimensionality reduction they also proposed a clustering method that worked over high dimensional data. Once the global principal components were obtained by using the distributed method and transmitted to each node, local data was projected onto the components and then a traditional clustering technique was applied. Finally, a central node integrated the local clusters in order to obtain a global clustering model.

[Li et al., 2003] proposed an algorithm named the *CoFD* algorithm, which was a non-distance based clustering algorithm for high dimensional spaces. Based on the Maximum Likelihood Principle, *CoFD* attempted to optimize its parameter settings to maximize the likelihood between data points and the model generated by the parameters. The distributed version of the algorithm, called *D-CoFD*, was also proposed. The authors claimed that the experimental results on both synthetic and real datasets showed the efficiency and effectiveness of *CoFD* and *D-CoFD* algorithms.

Several years later, [Liang et al., 2013] presented another algorithm for principal components extraction over distributed data. To this end, each node computed PCA over its local data and transmitted a fraction of them to a central node. This node used the received components to estimate the global principal components, which were later transmitted to each node. After this, in every node, local data were projected onto the global components and the projected data were used for computing a coreset by means of a distributed algorithm. The global coreset built from the local projected data was finally used to obtain a global clustering model.

[De Vries et al., 2015] 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 (a binary signature for each document). It is applied to two WEB scale crawls covering tens of terabytes (ClueWeb09 and ClueWeb12). The proposed algorithm employed the entire collection for clustering and produced several orders of magnitude more clusters than the existing algorithms. Fine grained clustering is necessary for meaningful clustering in massive collections where the number of distinct topics grows linearly with collection size. Fine-grained clusters showed an improved cluster quality when assessed with two novel evaluations using ad-hoc search relevance judgments and Spam classifications for external validation. These evaluations solved the problem of assessing the quality of clusters where categorical labeling was unavailable.

3.3 Distributed approaches for clustering based on density estimators

[Klusch et al., 2003] proposed a novel distributed clustering algorithm based on non-parametric kernel density estimation, which took into account the issues of privacy and communication costs in a low dimensional distributed data environment.

[Januzaj et al., 2004] presented a scalable version of *DBSCAN* that was also capable of operating over distributed collections. First, the best local representatives were selected depending on the number of points that each one represented and then those chosen points were sent to a central node. The central node clustered the local representatives into a single new model which was transmitted back to the other nodes to improve their local group structure. Unfortunately, only experiments performed over low dimensional data were reported.

3.4 Distributed approaches for clustering based on prototypes

[Forman and Zhang, 2000] described a technique to parallelize a family of center-based data clustering algorithms. The central idea of the proposed algorithm is to communicate only sufficient statistics, yielding linear speed-up with good efficiency. The

authors demonstrated 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., 2013] provided novel extensions in a distributed scenario for two popular prototype-based strategies, namely k-median and K-Means. The proposed algorithms reduced the problem of finding a clustering with low cost to the problem of finding a coreset of small size. The authors provided a distributed method for constructing a global coreset which represented an interesting improve over the previous methods due to the reduction in terms of the communication complexity. Experimental results on large scale datasets showed that this approach is feasible and competitive against centralized techniques.

3.5 Distributed approaches for clustering based on parametric models

[Merugu and Ghosh, 2003] presented 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 portions of the original data, the proposed method transmitted to a central node the parameters of local generative models built at each site. They showed that the best representative of all the data is a certain "average" model, and empirically showed that this model could be approximated quite well by generating artificial samples from the underlying distributions by using Markov Chain Monte Carlo techniques and then fitting a combined global model with a chosen parametric form to these samples. Also a new measure that quantifies privacy based on information theoretic concepts was proposed, and it showed that decreasing privacy leads to a higher quality of the combined model. They provided empirical results on different data types to highlight the generality of the framework. The results showed that high quality distributed clustering can be achieved with little privacy loss and low communication cost.

[Kriegel et al., 2005] proposed a distributed model-based clustering algorithm that used *EM* for building local models based on mixtures of Gaussian distributions. They presented an efficient and effective algorithm for deriving and merging local Gaussian distributions producing a meaningful global model. The experimental evaluation performed demonstrated that the framework scaled-up in a highly distributed environment.

3.6 Distributed approaches for Hierarchical clustering

A hierarchical algorithm, called Collective Hierarchical Clustering (CHC), that works on distributed and heterogeneous data was presented by [Johnson and Kargupta, 2000]. This algorithm first generated local cluster models and then combined them to generate the global cluster model of the data. The proposed algorithm ran in $O(|S|n^2)$ time, with a $O(|S|n)$ space requirement and $O(n)$ communication requirement, where n is the number of elements in the dataset and $|S|$ is the number of data sites. This approach showed a significant improvement over naive methods with $O(n^2)$ communication costs in the case that the entire distance matrix is transmitted and $O(nm)$ communication costs to centralize the data, where m is the total number of features. A specific implementation based on the single link clustering and results comparing its performance with that of a centralized clustering algorithm were presented. Additionally, an analysis of the algorithm complexity, in terms of overall computation time and communication requirements, was presented.

[Jin et al., 2015] proposed a hierarchical clustering algorithm for distributed data that built the clusters by incrementally processing the data points. The authors re-stated the hierarchical clustering problem as a Minimum Spanning Tree construction problem over a graph. In order to integrate several local models they also proposed a technique for combining multiple Minimum Spanning Trees, assuming that these trees were obtained from disjoint subgraphs of the complete original graph. This mixture procedure iterated until a single tree was obtained, which corresponded to the hierarchical clustering originally pursued.

3.7 Discussion

There are several approaches in the literature for the construction of clustering techniques capable of operating in scenarios where the data is distributed. Unfortunately, most of them have been focused on the large number of data points but with low dimensional computational representations (less than 100 attributes) as in the cases of [Klusch et al., 2003] and [Januzaj et al., 2004]. This contrasts to large document collections in which a document vector may have about 10^4 attributes. For instance the RCV1 dataset processed by using the Scikit-learn framework[3] contained 804414 documents computationally represented as vectors spanned onto a space of 47236 terms. Considering 4 Bytes to represent *Float* values and without resorting to any sparse representation, the space required for loading this entire matrix in main memory is about 140 Gigabytes which is beyond the RAM available in most computers.

The increasing storage cost is not the only issue since as dimensionality grows traditional distance measures such as Cosine and Euclidean become more uniform, making clustering more difficult [Ertöz et al., 2003]. Hierarchical and K-Medoids based algorithms, such as [Johnson and Kargupta, 2000] and [Jin et al., 2015], are specially affected by this problem. Additionally, algorithms enhanced by the MapReduce framework, such as [Ene et al., 2011], address the problem of processing a large number of documents but high dimensionality still affects each node of computation.

[3] `http://scikit-learn.org/stable/datasets/rcv1.html`

Some previous works deal with high dimensional data by using parallel approaches as in the case of [De Vries et al., 2015]. Besides the reduction in space requirements, the dataset is still limited by the available disk and RAM capacity of a single machine.

Considering the parallel processing improvements for centralized computation and the proposed strategies for distributed data processing made by previous contributions, the proposal presented in this project is related to two of the previously mentioned research trends, namely 3.3 and 3.4.

4 Proposal

We propose a Distributed and Parallel strategy for dealing with high dimensional and massive text data clustering based on the Shared Nearest Neighbors scheme (SNN) [Jarvis and Patrick, 1973][Ertöz et al., 2003]. We expect that the inspiration in these two succesfully employed centralized approaches allows firstly, to build more robust algorithms for high dimensional data and secondly, to build scalable distributed algorithms based on representative points found in each node.

The proposed framework uses a Master/Slave model of computation in which a master node partitions the dataset and then each slave or worker node processes its assigned chunk of data in order to identify points that represents each group (*Dataset partitioning* and *Local model generation* stages in Figure 1). Then, these selected points are transmitted to the Master node in order to build an overall model which is then transmitted back to the workers to perform a model refining procedure (*Model refining* dotted loop in Figure 1). This step can be performed several times, after which the group assignments are transmitted from each worker to the Master to generate the final result.

In addition to the distributed computation scheme, within each node it is possible to improve its performance by exploiting the underlying multicore technologies, e.g. in distance and k-nearest neighbor computation [Gavahi et al., 2015]. It is in this part of the proposed strategy where we expect that the introduction of parallel technologies such as GPU enhance the overall performance of the distributed algorithm.

The distributed computation strategy allows to decrease the cost of computing the complete dataset in one node which sometimes is unfeasible (Dataset sizes beyond the primary memory capacity). Furthermore, as each node can incorporate parallel routines (enabled by exploiting multicore processors or GPUs) to accelerate its partial computations the overall time cost of the algorithm is also improved.

5 Hypothesis

A Master/Slave Clustering method along with a Shared-Nearest-Neighbor strategy for the identification of representative points in each worker, will outperform other state of the art distributed approaches (in terms of clustering quality measures such as V-measure and Rand-Index), over text collections in which documents are distributed across several nodes in a network and whose total size exceeds current RAM available in modern desktop computers.

- The computational procedure performed within each node will exploit modern parallel technologies existing in multicore-architectures and the usage of GPU accelerated computing.

6 Goals

The general aim of this work consists in developing new parallel clustering techniques capable of dealing with large document databases that also can be stored in several nodes of a computer network. Additionally, this work comprises the following specific objectives:

- To develop parallel clustering scheme for document collections that also performs a single-pass over each document, i.e. access to secondary memory during the algorithm execution is restricted.
- To extend the parallel scheme proposed to environments where the document collection is distributed into several machines.
- To validate the proposed models with benchmark and real text data extracted from specialized sites such as UCI and NIST.
- To disseminate the obtained results and methods in this research in scientific journals registered in the ISI catalog.

7 Methodology

In order to acomplish the goals of this project the following general and specific activities are distinguished:

1. Constant revision and discussion of the state of the art literature related to:

- Parallel and distributed Clustering methods.
- Theoretical and experimental aspects involved in the single-pass and parallel clustering methods for distributed data.
- Hashing strategies for the efficient estimation of neighborhood over centralized and distributed data collections.
- Clustering methods for massive data collections.
- Distributed and parallel Machine Learning algorithms.

2. Design and construction of the proposed methods. In addition and considering the previously posed hypothesis and its validation, the following steps are established:
   - Modeling and analysis of results under environments presenting distributed and high dimensional data.
   - Construction of the proposed models. Development of distributed processing scheme for single-pass Clustering algorithm capable of operating under scenarios in which the vectors, each one representing a document, are distributed into several computers (workers).
   - An empirical study of techniques for summarizing massive text volumes (e.g. Coresets, medoids) with the aim to generate more succint snapshots of the current state of the stream. This is of particular interest in a distributed scenario in which a message passing based synchronization is performed between workers and the communication cost is restrictive.
   - Analysis of the algorithmic complexity in terms of space and time of the proposed models.
3. During the development of the algorithm implementations we identified the following steps: (a) Construction of a modular framework adequately general to serve as a basis for the posterior construction of the proposal and its variants. (b) The implementation and testing of several distributed scenarios (e.g. Balanced and Imbalance load in workers, arrival of new instances and its posterior labeling and the incorporation of concept-drift by introducing new classes to the algorithm). (c) Implementation and evaluation of baseline algorithms of the state of the art (e.g. Spark K-Means, Spark-DBSCAN and RACHET [[Samatova et al., 2002]]).
4. Design of the experiments and validation of the proposed algorithms. In order to validate the proposed techniques, we plan to use benchmark datasets coming from several sources; namely, the UCI machine learning repository[4], the Tipster text collection[5] and the GOV2 text collection[6]. In some cases the document set is labeled or at least some labels can be inferred from relevance judges (e.g. the Tipster collection), and in some other no label information is available (e.g. GOV2 collection). There exist several measures for the quantification of clustering quality. When labeling information is available external measures such as V-measure [[Rosenberg and Hirschberg, 2007]] and Adjusted Rand Index [[Hubert and Arabie, 1985]] are applied. In the absence of this information, internal measures such as the Silhouette [[Rousseeuw, 1987]] coefficient and CVNN [[Liu et al., 2013]] are utilized. As the partitioning of the data is performed at random, the validation will be conducted by executing 20 runs over each dataset. Then, the average and standard deviation of each attained quality measure will be reported.
5. Colaboration and dissemination activities:
   - Seminar with topics about Clustering and Non-Supervised methods and their applications over problems involving distributed and massive data collections. Researchers, practitioners and students (undergraduate and postgraduate) will be included.
   - Participation in national and international conferences related to Machine Learning techniques, Intelligent Systems for distributed data and Knowledge extraction from large databases (at least one per year).
   - Establishing contact for cooperation with related research groups either in the national as in the international sphere.

8 Work Plan

The stages were explained in the Methodology section and are the following:

- *Stage 1*: Study and Discussion of the relevant literature for the proposal.
- *Stage 2*: Design of the proposed models and algorithms:
  1. Design of a processing strategy for distributed data clustering.

[4] http://archive.ics.uci.edu/ml/index.php
[5] https://catalog.ldc.upenn.edu/LDC93T3A
[6] http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm

2. Implementation of the proposed scheme.

- *Stage 3*: Design of Experiments and validation of the proposed algorithms:
  1. Synthetic data generation.
  2. Processing real text data. For the evaluation of clustering algorithms a general practice consists in using datasets employed for classification since these have labeling information, hence external evaluation measures for Clustering can be used. Additionally, some datasets not labeled but large in size, such as GOV2, will be considered.
  3. Comparative analysis against state of the art techniques. External and internal measures will be used and the achieved values will be compared against the ones attained by techniques such as DBSCAN and K-Means implemented within the MapReduce framework for distributed computing.
- *Stage 4*: Dissemination of the attained results.
- *Stage 5*: Study of algorithmic complexity in terms of time and space.
  - Exploration of potential optimizations of the proposed algorithms.
  - Discussion about the scalability of the proposed algorithms.
- *Stage 6*: Integration of the proposals with GPU computation at the Master and Workers levels. In this stage, GPU will be specially used to improve array products for distance computation.
- *Stage 7*: Final dissemination of the attained results and contributions of this project.

In summary, during year 2018, we will concentrate on the development of a distributed framework that addresses the high dimensional data clustering task from a core-representatives approach. That is, the overall data clustering is obtained from representatives chosen independently in each local worker (S1). Therefore the effort will be put on techniques based on Coresets and distances derived from Nearest-Neighbors shared between data points (S2). In order to assess the performance of the proposal, real high dimensional datasets will be used and a thorough experimental validation will be made (measuring clustering quality through several executions of the algorithm and under different parameter settings) (S3). At the end of 2018, the built algorithms together with their attained results will be reported in at least one indexed journal (S4).

During the year 2019, we will focus firstly, on the study of the algorithmic complexity of the overall distributed framework along with the complexity of the tasks performed by each worker (S5). Lastly, the effort will be put on the usage of GPU computation to enhance performance times specially on the computation of distances and neighborhoods (these are the two most expensive operations) (S6). The improvements and findings obtained during this year will be published in at least one indexed journal (S7).

9 Work in progress

The Shared-Nearest-Neighbor clustering algorithm presented by [Ertöz et al., 2003] proposes the integration of the DBSCAN clustering algorithm along with a shared-neighbors based similarity measure. This scheme tackles the variable density, size among clusters and also the curse of dimensionality in a centralized setting. In spite of its valuable contribution this method assumes that pairwise distance/similarity computation and storage is always feasible. Due to the quadratic time complexity involved in the computation of the distances and the similar space complexity to store these data, centralized methods that rely on this calculations do not scale in massive data scenarios.

An initial approach that we took is to explore the construction of clustering algorithms capable of dealing with high dimensional and distributed text collections by adapting the centralized algorithm proposed by [Ertöz et al., 2003]. Consequently, the curse of dimensionality is addressed by treating the distance between two data points as a function of the nearest neighbors they share (nearest in terms of a traditional metric such as the euclidean distance). Additionally, the large number of observations and the potential noise or abnormal data points are addressed by a representatives selection mechanism performed on each worker. Is in this way that each worker reports only those points from its local subset that share a sufficient quantity of neighbors with their nearest neighbors. The rationale behind this scheme is that the presence of noise is denoted by more or less isolated data points. Then, the master node concentrates all these core points, builds a network based on their shared-nearest-neighbor similarity and labels them by using a network propagation scheme such as the one proposed by [Raghavan et al., 2007]. Finally, those core points along with their labels are transmitted back to the worker nodes, each worker labels its local subset by using the label of its nearest corepoint and then retransmits this information to the master node. Once the master node receives all the labels it reports the final clustering of the collection.

The ongoing work described above has been tested on synthetic and real text data but with reduced size. An accepted article at the *CIARP-2017*[7] conference shows some preliminary results attained with the proposed scheme. The article is titled *"A Distributed Shared Nearest Neighbors Clustering Algorithm"* and is authored by the principal researcher, the sponsoring researcher and a coleague from the *Universidad Técnica Federico Santa María*.

[7] www.ciarp2017.org

We expect to be able to test the algorithm with really large collections (more than several hundred of thousands of documents) and also to improve the local workers by exploiting the power of GPU computing.

10 Available Resources

The resources available for this project at the *Escuela de Ingeniera Informtica* of the *Pontificia Universidad Catlica de Valparaso* are:

- University library, free access to electronic libraries such as IEEE and Springer and the personal library of the sponsoring researcher at the previously mentioned campus.
- A room for conferences.
- An office with telephone and fast Internet connection.
- A Dell R730 server (20 physical cores and 128 GB RAM).

Open access to programming languages such as Julia, Python, Java, C++ and R, and to the Latex document preparation system.

References

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. (2012). Scalable K-Means++. Proceedings of the VLDB Endowment (PVLDB), 5:622–633.

Balcan, M. F., Ehrlich, S., and Liang, Y. (2013). Distributed K-Means and K-Median Clustering on General Topologies. Advances in Neural Information Processing Systems 26 (NIPS 2013), pages 1–9.

Crestani, F., and Markov, I. (2013). Distributed Information Retrieval and Applications. 35th European Conference on IR Research, 865–868.

Das, A., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th international conference on World Wide Web, pages 271–280. ACM.

De Vries, C. M., De Vine, L., Geva, S., and Nayak, R. (2015). Parallel streaming signature em-tree: A clustering algorithm for web scale applications. In Proceedings of the 24th International Conference on World Wide Web, pages 216–226. ACM.

De Vries, C. M. and Geva, S. (2009). K Tree: large scale document clustering. Proceedings of the 32nd international ACM SIGIR conference on Research and development in Information Retrieval, pp. 718–719, ACM.

Dhillon, I. S. and Modha, D. S. (1999). A data-clustering algorithm on distributed memory multiprocessors. LargeScale Parallel Data Mining, 1759(802):245–260.

Ene, A., Im, S., Moseley, B. (2011). Fast Clustering using MapReduce. Kdd, 681–689.

Ertöz, L., Steinbach, M., and Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. Proceedings of the SIAM International Conference on Data Mining, 47–58.

Forman, G. and Zhang, B. (2000). Distributed data clustering can be efficient and exact. ACM SIGKDD Explorations Newsletter, 2(2):34–38.

Gavahi, M., Mirzaei, R., Nazarbeygi, A., Ahmadzadeh, A. and Gorgin, S. (2015). High performance GPU implementation of k-NN based on Mahalanobis distance. International Symposium on Computer Science and Software Engineering, 1–6.

Han, J., Pei, J., and Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

Hubert, L. and Arabie, P. (1985). Comparing partitions. Journal of Classification, 2:1, 193–218, Springer.

Jagannathan, G., and Wright, R. N. (2005). Privacy-preserving Distributed K-Means Clustering over Arbitrarily Partitioned Data. Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 593–599.

Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2003). Towards Effective and Efficient Distributed Clustering. Workshop on Clustering Large Data Sets, pages 49–58.

Januzaj, E., Kriegel, H.-P., and Pfeifle, M. (2004). Scalable Density-Based Distributed Clustering. pages 231–244.

Jarvis, R.A. and Patrick, E.A. (1973). Clustering Using a Similarity Measure Based on Shared Near Neighbors. IEEE Transactions on Computers, 22:1025–1034.

Jin, C., Chen, Z., Hendrix, W., Agrawal, A., and Choudhary, A. (2015). Incremental, Distributed Single-linkage Hierarchical Clustering Algorithm Using Mapreduce. Proceedings of the Symposium on High Performance Computing, pages 83–92.

Johnson, E. and Kargupta, H. (2000). Collective, hierarchical clustering from distributed, heterogeneous data. Lecture Notes in Computer Science, 1759:221–244.

Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E. (2001). Distributed clustering using collective principal component analysis. Knowledge and Information Systems, 3(4):422–448.

Klusch, M., Lodi, S., and Moro, G. (2003). Distributed clustering based on sampling local density estimates. IJCAI International Joint Conference on Artificial Intelligence, pages 485–490.

Kriegel, H.-p., Kr, P., Pryakhin, A., and Schubert, M. (2005). Effective and Efficient Distributed Model-based Clustering.

Li, T., Zhu, S., and Ogihara, M. (2003). Algorithms for Clustering High Dimensional and Distributed Data. Intelligent Data Analysis Journal, 7(February):1–36.

Liang, Y., Balcan, M.-f., and Kanchanapally, V. (2013). Distributed PCA and K-Means Clustering. The Big Learning Workshop in NIPS 2013, pages 1–8.

Liu, J., Huang, J. Z., Luo, J., and Xiong, L. (2012). Privacy Preserving Distributed DBSCAN Clustering. Proceedings of the 2012 Joint EDBT/ICDT Workshops, 177–185.

Liu, Y., Li, Z. Xiong, H., Gao, X., Wu, J. and Wu, S. (2013). Understanding and Enhancement of Internal Clustering Validation Measures. IEEE Transactions on Cybernetics, 43:3, pp. 982–994, June 2013.

Merugu, S. and Ghosh, J. (2003). Privacy-preserving Distributed Clustering using Generative Models. Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM), pp. 0–7.

Nagwani, N. K. (2015). Summarizing large text collection using topic modeling and clustering based on MapReduce framework. Journal of Big Data, 2:1–18.

Naldi, M. C. and Campello, R. J. G. B. (2014). Evolutionary K-Means for distributed data sets. Neurocomputing, 127:30–42.

Nayak, R. and Mills, R. and De Vries, C. and Geva, S. (2014). Clustering and Labeling a Web Scale Document Collection using Wikipedia clusters Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval, pp. 23–30, ACM.

Raghavan, U., N., Albert, R. and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Physical review, 76(3).

Rosenberg, A. and Hirschberg, J. (2007). V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.. Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 7:410–420.

Rousseeuw Peter J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65.

Samatova, N. F. and Ostrouchov, G. and Geist, A. and Melechko, A. V. (2002). RACHET: An Efficient Cover-Based Merging of Clustering Hierarchies from Distributed Datasets. Journal of Distributed and Parallel Databases, 11:2, 157–180, Springer.

Sanders, J. and Kandrot, E. (2010). CUDA by Example: An Introduction to General-Purpose GPU Programming (1st edition). Addison-Wesley Professional.

Xu, X., Jäger, J., and Kriegel, H. (2002). A fast parallel clustering algorithm for large spatial databases. High Performance Data Mining, 290:263–290.

Qi, Z., Jinze, L., and Wei, W. (2008). Approximate clustering on distributed data streams. Proceedings - International Conference on Data Engineering, 00:1131–1139.